



# Синтаксис скрипта и функции диаграммы

---

Qlik Sense®

3.2

Copyright © 1993-2017 QlikTech International AB. Все права защищены.



© QlikTech International AB 1993-2017. Все права защищены.

Qlik®, QlikTech®, Qlik Sense®, QlikView®, Sense® и логотип Qlik являются товарными знаками компании QlikTech International AB, зарегистрированными во многих странах. Другие товарные знаки являются товарными знаками соответствующих владельцев.

<b>1</b>	<b>Что такое Qlik Sense?</b>	<b>19</b>
1.1	Что можно сделать с помощью программы Qlik Sense?	19
1.2	Как работает программа Qlik Sense?	19
	Модель приложения	19
	Ассоциативная работа	19
	Совместная работа и мобильность	20
1.3	Как развернуть программу Qlik Sense?	20
	Qlik Sense Desktop	20
	Qlik Sense Enterprise	20
1.4	Как осуществлять контроль и управление сайтом Qlik Sense	20
1.5	Расширение возможностей Qlik Sense и адаптация под ваши требования	20
	Построение расширений и гибридных веб-приложений	20
	Построение клиентов	21
	Построение инструментов сервера	21
	Подключение к другим источникам данных	21
<b>2</b>	<b>Синтаксис скрипта</b>	<b>22</b>
2.1	Введение в синтаксис скрипта	22
2.2	Что такое форма Backus-Naur?	22
2.3	Операторы и ключевые слова скрипта	23
	Операторы управления скриптом	24
	Обзор операторов управления скриптом	24
	Call	26
	Do..loop	27
	Exit script	28
	For..next	28
	For each..next	30
	If..then..elseif..else..end if	32
	Sub..end sub	34
	Switch..case..default..end switch	35
	Префиксы скрипта	36
	Обзор префиксов скрипта	36
	Add	40
	Buffer	41
	Concatenate	42
	Crosstable	43
	First	44
	Generic	44
	Hierarchy	45
	HierarchyBelongsTo	46
	Inner	48
	IntervalMatch	49
	Join	53
	Keep	53
	Left	54

Mapping .....	56
NoConcatenate .....	57
Outer .....	58
Replace .....	59
Right .....	60
Sample .....	62
Semantic .....	62
Unless .....	63
When .....	63
Обычные операторы скриптов .....	64
Обзор обычных операторов скриптов .....	64
Alias .....	71
Binary .....	71
Comment field .....	72
Comment table .....	73
Connect .....	74
Declare .....	76
Установка нового определения поля .....	76
Повторное использование существующего определения поля .....	77
Derive .....	78
Direct Query .....	79
Списки полей Direct Discovery .....	82
Directory .....	84
Disconnect .....	85
Drop field .....	85
Drop table .....	86
Execute .....	87
FlushLog .....	88
Force .....	89
Load .....	90
Элементы спецификации формата .....	99
Набор символов .....	99
Формат таблицы .....	100
Delimiter is .....	101
No eof .....	101
Labels .....	102
Header is .....	102
Record is .....	103
Quotes .....	104
XML .....	104
KML .....	105
Let .....	105
Loosen Table .....	105
Map .....	106
NullAsNull .....	107

NullAsValue .....	108
Qualify .....	108
Rem .....	109
Rename field .....	110
Rename table .....	111
Search .....	111
Section .....	112
Select .....	113
Set .....	115
Sleep .....	116
SQL .....	116
SQLColumns .....	117
SQLTables .....	118
SQLTypes .....	118
Star .....	119
Store .....	121
Tag .....	122
Trace .....	123
Unmap .....	123
Unqualify .....	124
Untag .....	125
Рабочий каталог .....	125
Рабочий каталог Qlik Sense Desktop .....	125
Рабочий каталог Qlik Sense .....	126
2.4 Работа с переменными в редакторе загрузки данных .....	126
Обзор .....	126
Определение переменной .....	126
Удаление переменной .....	127
Загрузка значения переменной в качестве значения поля .....	127
Вычисление переменной .....	127
Системные переменные .....	128
Обзор системных переменных .....	128
CreateSearchIndexOnReload .....	131
HidePrefix .....	132
HideSuffix .....	132
Include .....	133
OpenUrlTimeout .....	133
StripComments .....	134
Verbatim .....	134
Значение, обрабатывающее переменные .....	134
Обзор значений, обрабатывающих переменные .....	134
NullDisplay .....	135
NullInterpret .....	135
NullValue .....	136

OtherSymbol .....	136
Переменные интерпретации числа .....	136
Обзор переменных интерпретации числа .....	137
Форматирование валюты .....	137
Формат чисел .....	137
Форматирование времени .....	138
BrokenWeeks .....	139
DateFormat .....	140
DayNames .....	140
DecimalSep .....	140
FirstWeekDay .....	140
LongDayNames .....	141
LongMonthNames .....	141
MoneyDecimalSep .....	141
MoneyFormat .....	142
MoneyThousandSep .....	142
MonthNames .....	142
ReferenceDay .....	142
ThousandSep .....	143
TimeFormat .....	143
TimestampFormat .....	143
Переменные Direct Discovery .....	144
Системные переменные Direct Discovery .....	144
Переменные чередования запросов Teradata .....	145
Символьные переменные Direct Discovery .....	146
Переменные интерпретации числа Direct Discovery .....	147
Ошибка переменных .....	148
Обзор ошибок переменных .....	148
ErrorMode .....	149
ScriptError .....	149
ScriptErrorCount .....	150
ScriptErrorList .....	151
2.5 Выражения скрипта .....	151
<b>3 Выражения визуализации .....</b>	<b>153</b>
3.1 Определение объема агрегирования .....	153
3.2 Синтаксис для множеств .....	155
3.3 Модификаторы множества .....	156
На основе другого поля .....	156
На основе множеств элементов (список значений поля в модификаторе) .....	156
Принудительное исключение .....	157
Модификаторы множества с операторами множества .....	158
Модификаторы множества, использующие назначения с операторами множества	
implicit .....	159
Модификаторы множества с расширенным поиском .....	159

Модификаторы множества с расширениями со знаком доллара .....	160
Модификаторы множества с определениями значений поля implicit .....	160
3.4 Выражение визуализации и синтаксис агрегирования .....	161
Общий синтаксис выражений диаграммы .....	161
Общий синтаксис для агрегирования .....	162
<b>4 Операторы .....</b>	<b>163</b>
4.1 Побитовые операторы .....	163
4.2 Логические операторы .....	164
4.3 Числовые операторы .....	164
4.4 Реляционные операторы .....	165
4.5 Строковые операторы .....	166
<b>5 Функции в скриптах и выражениях диаграммы .....</b>	<b>168</b>
5.1 Функции агрегирования .....	168
Использование функций агрегирования в скрипте загрузки данных .....	168
Использование функций агрегирования в выражениях диаграмм .....	168
Aggr — функция диаграммы .....	169
Базовые функции агрегирования .....	173
Обзор базовых функций агрегирования .....	173
Базовые функции агрегирования в скрипте загрузки данных .....	173
Базовые функции агрегирования в выражениях диаграмм .....	174
FirstSortedValue .....	175
FirstSortedValue — функция диаграммы .....	177
Max .....	179
Max — функция диаграммы .....	180
Min .....	183
Min — функция диаграммы .....	184
Mode .....	187
Mode — функция диаграммы .....	188
Only .....	189
Only — функция диаграммы .....	190
Sum .....	192
Sum — функция диаграммы .....	193
Функции агрегирования счетчика .....	195
Функции агрегирования счетчика в скрипте загрузки данных .....	195
Функции агрегирования счетчика в выражениях диаграмм .....	196
Count .....	197
Count — функция диаграммы .....	198
MissingCount .....	200
MissingCount — функция диаграммы .....	202
NullCount .....	204
NullCount — функция диаграммы .....	205
NumericCount .....	206
NumericCount — функция диаграммы .....	208
TextCount .....	210

TextCount — функция диаграммы .....	211
Функции финансового агрегирования .....	213
Функции финансового агрегирования в скрипте загрузки данных .....	213
Функции финансового агрегирования в выражениях диаграмм .....	214
IRR .....	215
IRR — функция диаграммы .....	216
NPV .....	217
NPV — функция диаграммы .....	219
XIRR .....	220
XIRR — функция диаграммы .....	221
XNPV .....	222
XNPV — функция диаграммы .....	224
Функции статистического агрегирования .....	225
Функции статистического агрегирования в скрипте загрузки данных .....	226
Функции статистического агрегирования в выражениях диаграмм .....	228
Avg .....	232
Avg — функция диаграммы .....	233
Correl .....	235
Correl — функция диаграммы .....	236
Fractile .....	238
Fractile — функция диаграммы .....	239
Kurtosis .....	241
Kurtosis — функция диаграммы .....	243
LINEST_B .....	244
LINEST_B — функция диаграммы .....	245
LINEST_DF .....	247
LINEST_DF — функция диаграммы .....	247
LINEST_F .....	249
LINEST_F — функция диаграммы .....	250
LINEST_M .....	251
LINEST_M — функция диаграммы .....	252
LINEST_R2 .....	253
LINEST_R2 — функция диаграммы .....	254
LINEST_SEB .....	255
LINEST_SEB — функция диаграммы .....	256
LINEST_SEM .....	258
LINEST_SEM — функция диаграммы .....	258
LINEST_SEY .....	260
LINEST_SEY — функция диаграммы .....	261
LINEST_SSREG .....	262
LINEST_SSREG — функция диаграммы .....	263
LINEST_SSRESID .....	264
LINEST_SSRESID — функция диаграммы .....	265
Median .....	267
Median — функция диаграммы .....	268



Skew .....	269
Skew — функция диаграммы .....	270
Stdev .....	272
Stdev — функция диаграммы .....	273
Sterr .....	275
Sterr — функция диаграммы .....	276
STEYX .....	278
STEYX — функция диаграммы .....	279
Примеры использования функций linest .....	281
Загрузка данных образца .....	281
Отображение результатов из вычислений скрипта загрузки данных .....	282
Создание визуализаций функции диаграммы linest .....	283
Статистические функции тестирования .....	284
Функции критерия Хи-квадрат .....	284
Функции Т-критериев .....	284
Функции Z-критериев .....	284
Функции критерия Хи-квадрат .....	284
Chi2Test_chi2 .....	285
Chi2Test_df .....	286
Chi2Test_p — функция диаграммы .....	287
Функции Т-критериев .....	288
TTest_conf .....	292
TTest_df .....	293
TTest_dif .....	294
TTest_lower .....	295
TTest_sig .....	296
TTest_sterr .....	297
TTest_t .....	298
TTest_upper .....	299
TTestw_conf .....	300
TTestw_df .....	301
TTestw_dif .....	302
TTestw_lower .....	303
TTestw_sig .....	305
TTestw_sterr .....	306
TTestw_t .....	307
TTestw_upper .....	308
TTest1_conf .....	309
TTest1_df .....	310
TTest1_dif .....	311
TTest1_lower .....	311
TTest1_sig .....	312
TTest1_sterr .....	313
TTest1_t .....	314
TTest1_upper .....	315

TTest1w_conf	316
TTest1w_df	317
TTest1w_dif	317
TTest1w_lower	318
TTest1w_sig	319
TTest1w_sterr	320
TTest1w_t	321
TTest1w_upper	322
Функции Z-критериев	323
ZTest_z	325
ZTest_sig	326
ZTest_dif	327
ZTest_sterr	328
ZTest_conf	329
ZTest_lower	330
ZTest_upper	331
ZTestw_z	332
ZTestw_sig	333
ZTestw_dif	334
ZTestw_sterr	335
ZTestw_conf	336
ZTestw_lower	337
ZTestw_upper	338
Примеры статистических тестовых функций	339
Примеры использования функций chi2-test в диаграммах	339
Примеры использования функций chi2-test в скрипте загрузки данных	342
Создание типичного отчета t-test	344
Примеры использования функций z-test	347
Строковые функции агрегирования	349
Строковые функции агрегирования в скрипте загрузки данных	349
Строковые функции агрегирования в диаграммах	350
Concat	351
Concat — функция диаграммы	352
FirstValue	354
LastValue	355
MaxString	356
MaxString — функция диаграммы	357
MinString	359
MinString — функция диаграммы	360
Функции синтетических измерений	362
ValueList — функция диаграммы	362
ValueLoop — функция диаграммы	364
Вложенные агрегирования	365
Вложенные агрегирования с префиксом TOTAL	365
5.2 Функции цвета	365

Предопределенные функции цвета .....	368
ARGB .....	369
RGB .....	369
HSL .....	370
5.3 Условные функции .....	371
Обзор условных функций .....	371
alt .....	372
class .....	373
if .....	374
match .....	375
mixmatch .....	375
pick .....	375
wildmatch .....	376
5.4 Функции счетчика .....	376
Обзор функций счетчика .....	376
autonumber .....	378
autonumberhash128 .....	380
autonumberhash256 .....	382
IterNo .....	385
RecNo .....	385
RowNo .....	386
RowNo — функция диаграммы .....	388
5.5 Функции даты и времени .....	389
Обзор функций даты и времени .....	390
Целочисленные выражения времени .....	390
Функции меток времени .....	391
Функции формирования .....	391
Другие функции даты .....	392
Функции часовых поясов .....	392
Функции установки времени .....	393
Функции вхождения .....	393
Функции начала и конца .....	395
Функции нумерации дней .....	398
addmonths .....	399
addyears .....	400
age .....	400
converttolocaltime .....	402
day .....	404
dayend .....	405
daylightsaving .....	406
dayname .....	406
daynumberofquarter .....	408
daynumberofyear .....	409
daystart .....	411

---

firstworkdate .....	412
GMT .....	414
hour .....	414
inday .....	415
indaytotime .....	417
inlunarweek .....	418
inlunarweektodate .....	420
inmonth .....	422
inmonths .....	424
inmonthstodate .....	426
inmonthtodate .....	428
inquarter .....	430
inquartertodate .....	432
inweek .....	434
inweektodate .....	435
inyear .....	437
inyeartodate .....	439
lastworkdate .....	441
localtime .....	443
lunarweekend .....	444
lunarweekname .....	445
lunarweekstart .....	447
makedate .....	449
maketime .....	450
makeweekdate .....	450
minute .....	451
month .....	451
monthend .....	452
monthname .....	453
monthsend .....	455
monthsname .....	457
monthsstart .....	459
monthstart .....	461
networkdays .....	463
now .....	464
quarterend .....	465
quartername .....	467
quarterstart .....	469
second .....	471
setdateyear .....	472
setdateyearmonth .....	473
timezone .....	474
today .....	475
UTC .....	475
week .....	476

---

weekday .....	478
weekend .....	479
weekname .....	481
weekstart .....	483
weekyear .....	485
year .....	486
yearend .....	486
yeamame .....	488
yearstart .....	490
yeartodate .....	492
5.6 Экспоненциальные и логарифмические функции .....	493
5.7 Функции поля .....	495
Функции счетчика .....	495
Функции поля и выборки .....	496
GetAlternativeCount — функция диаграммы .....	496
GetCurrentSelections — функция диаграммы .....	497
GetExcludedCount — функция диаграммы .....	498
GetFieldSelections — функция диаграммы .....	500
GetNotSelectedCount — функция диаграммы .....	501
GetPossibleCount — функция диаграммы .....	502
GetSelectedCount — функция диаграммы .....	503
5.8 Функции файлов .....	504
Обзор функций файла .....	504
Attribute .....	506
ConnectionString .....	514
FileName .....	514
FileDir .....	514
FileExtension .....	515
FileName .....	515
FilePath .....	515
FileSize .....	516
FileTime .....	517
GetFolderPath .....	517
QvdCreateTime .....	518
QvdFieldName .....	519
QvdNoOfFields .....	520
QvdNoOfRecords .....	521
QvdTableName .....	522
5.9 Финансовые функции .....	523
Обзор финансовых функций .....	524
BlackAndSchole .....	524
FV .....	525
nPer .....	526
Pmt .....	527

PV .....	528
Rate .....	529
5.10 Функции форматирования .....	530
Обзор функций форматирования .....	530
ApplyCodepage .....	532
Date .....	533
Dual .....	534
Interval .....	535
Money .....	536
Num .....	537
Time .....	539
Timestamp .....	539
5.11 Общие числовые функции .....	540
Обзор общих числовых функций .....	540
Функции сочетаний и перестановок .....	541
Функции Modulo .....	541
Функции четности .....	542
Функции округления .....	542
BitCount .....	542
Ceil .....	543
Combin .....	544
Div .....	545
Even .....	545
Fabs .....	545
Fact .....	546
Floor .....	546
Fmod .....	547
Frac .....	548
Mod .....	549
Odd .....	549
Permut .....	550
Round .....	550
Sign .....	552
5.12 Геопространственные функции .....	553
Обзор геопространственных функций .....	553
GeoAggrGeometry .....	554
GeoBoundingBox .....	556
GeoCountVertex .....	556
GeoGetBoundingBox .....	557
GeoGetPolygonCenter .....	557
GeoInvProjectGeometry .....	558
GeoMakePoint .....	559
GeoProject .....	559
GeoProjectGeometry .....	560

GeoReduceGeometry .....	561
5.13 Функции интерпретации .....	562
Обзор функций интерпретации .....	562
Date# .....	563
Interval# .....	564
Money# .....	565
Num# .....	566
Text .....	567
Time# .....	567
Timestamp# .....	568
5.14 Функции между записями .....	569
Функции строки .....	569
Функции столбца .....	570
Функции поля .....	571
Функции сводной таблицы .....	571
Межзаписные функции в скрипте загрузки данных .....	572
Above — функция диаграммы .....	573
Below — функция диаграммы .....	578
Bottom — функция диаграммы .....	582
Column — функция диаграммы .....	586
Dimensionality — функция диаграммы .....	588
Exists .....	589
FieldIndex .....	591
FieldValue .....	592
FieldValueCount .....	594
LookUp .....	595
NoOfRows — функция диаграммы .....	597
Peek .....	598
Previous .....	601
Top — функция диаграммы .....	602
SecondaryDimensionality — функция диаграммы .....	606
After — функция диаграммы .....	606
Before — функция диаграммы .....	608
First — функция диаграммы .....	609
Last — функция диаграммы .....	610
ColumnNo — функция диаграммы .....	610
NoOfColumns — функция диаграммы .....	611
5.15 Логические функции .....	612
5.16 Функции сопоставления .....	613
Обзор функций сопоставления .....	613
ApplyMap .....	613
MapSubstring .....	615
5.17 Математические функции .....	616
5.18 Функции NULL .....	617

Обзор функций NULL .....	617
IsNull .....	617
NULL .....	618
5.19 Функции над выборкой .....	619
Базовые функции над выборкой .....	620
Функции над выборкой счетчика .....	621
Статистические функции над выборкой .....	621
Финансовые функции над выборкой .....	622
RangeAvg .....	623
RangeCorrel .....	625
RangeCount .....	627
RangeFractile .....	629
RangeIRR .....	631
RangeKurtosis .....	632
RangeMax .....	633
RangeMaxString .....	635
RangeMin .....	637
RangeMinString .....	639
RangeMissingCount .....	640
RangeMode .....	642
RangeNPV .....	644
RangeNullCount .....	645
RangeNumericCount .....	646
RangeOnly .....	648
RangeSkew .....	649
RangeStdev .....	650
RangeSum .....	652
RangeTextCount .....	654
RangeXIRR .....	655
RangeXNPV .....	656
5.20 Функции ранжирования в диаграммах .....	657
Rank — функция диаграммы .....	658
HRank — функция диаграммы .....	662
5.21 Функции статистического распределения .....	664
Обзор функций статистического распределения .....	664
CHIDIST .....	665
CHIINV .....	666
FDIST .....	666
FINV .....	667
NORMDIST .....	668
NORMINV .....	668
TDIST .....	669
TINV .....	670
5.22 Строковые функции .....	671



Обзор строковых функций .....	671
Capitalize .....	674
Chr .....	674
Evaluate .....	675
FindOneOf .....	675
Hash128 .....	676
Hash160 .....	676
Hash256 .....	676
Index .....	677
KeepChar .....	677
Left .....	678
Len .....	679
Lower .....	679
LTrim .....	679
Mid .....	680
Ord .....	681
PurgeChar .....	681
Repeat .....	682
Replace .....	682
Right .....	683
RTrim .....	683
SubField .....	684
SubStringCount .....	686
TextBetween .....	687
Trim .....	687
Upper .....	688
5.23 Системные функции .....	688
Обзор системных функций .....	688
GetObjectField — функция диаграммы .....	691
IsPartialReload .....	691
ProductVersion .....	691
StateName — функция диаграммы .....	691
5.24 Функции таблиц .....	692
Обзор функций таблицы .....	692
FieldName .....	694
FieldNumber .....	694
NoOfFields .....	695
NoOfRows .....	695
5.25 Тригонометрические и гиперболические функции .....	696
<b>6 Ограничение доступа к файловой системе .....</b>	<b>698</b>
6.1 Аспекты безопасности при подключении к файлу на основе подключений данных .....	
ODBC и OLE DB .....	698
6.2 Ограничения в стандартном режиме .....	698
Системные переменные .....	699

---

Обычные операторы скриптов .....	701
Операторы управления скриптом .....	703
Функции файлов .....	703
Системные функции .....	707
6.3 Отключение стандартного режима .....	707
Qlik Sense .....	707
Qlik Sense Desktop .....	707
<b>7 Функции и операторы QlikView, не поддерживаемые в Qlik Sense .....</b>	<b>709</b>
7.1 Операторы скрипта, не поддерживаемые в Qlik Sense .....	709
7.2 Функции, не поддерживаемые в Qlik Sense .....	709
7.3 Префиксы, не поддерживаемые в Qlik Sense .....	709
<b>8 Функции и операторы, не рекомендуемые в Qlik Sense .....</b>	<b>711</b>
8.1 Операторы скрипта, не рекомендуемые в Qlik Sense .....	711
8.2 Параметры оператора скрипта, не рекомендуемые в Qlik Sense .....	711
8.3 Функции, не рекомендуемые в Qlik Sense .....	713
Префикс ALL .....	713

# 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 осуществлять пользователю 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 Синтаксис 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:

	Логическая операция OR: символ можно использовать с любой стороны.
( )	Скобки очередности выполнения: используются для структурирования синтаксиса BNF.
[ ]	Квадратные скобки: заключенные в них элементы являются необязательными.
{ }	Фигурные скобки: заключенные в них элементы могут повторяться ноль и более раз.

<b>Символ</b>	Нетерминальная синтаксическая категория: может быть разделена на другие символы. Например на составляющие вышеуказанного, другие нетерминальные символы, текстовые строки и т. д.
<b>::=</b>	Отметка начала блока, определяющего символ.
<b>LOAD</b>	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 };
```

## 2.3 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.

### Операторы управления скриптом

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

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:

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.

### 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
  End For
```

```

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:

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

### Пример:

```

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

```

### 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:

Аргумент	Описание
condition	Логическое выражение, имеющее значение True или False.
when / unless	Оператор <b>exit script</b> можно сделать условным с помощью дополнительного использования предложения <b>when</b> или <b>unless</b> .

#### 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:

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

### Пример 1: Загрузка последовательности файлов

```
// LOAD files file1.csv..file9.csv
for a=1 to 9
    LOAD * from file$(a).csv;
next
```

### Пример 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:

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

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
```

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

### Пример 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
```

### Пример 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://myData')
```

### Пример 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:
```

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.

См.: *if (страница 374)* (функция скрипта и диаграммы)



### 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:

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

### Пример 1:

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

### Пример 2:

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

### Пример 3:

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

## 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.

Поскольку оператор **sub** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из двух его возможных выражений (**sub** и **end sub**) не должно выходить за границу строки.

### 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.

### Пример 1:

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

### Пример 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 А будет инициализировано как 1, В как 4 и С как NULL.

При выходе из подпрограммы глобальная переменная А получает значение 2 (скопированное из подпрограммы). Второй фактический параметр  $(X+1)*2$  не будет копироваться, поскольку не является переменной. Наконец, глобальная переменная С 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:

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 <b>case</b> , то будут выполнены операторы в выражении <b>default</b> 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
```

### 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** можно добавить в любой оператор **LOAD**, **SELECT** или **map...using** в скрипте. Он применяется только во время частичной загрузки.

```
Add [only] (loadstatement | selectstatement | mapstatement)
```

#### Buffer

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

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

#### Concatenate

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

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

#### Crosstable

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

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

#### First

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

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

### Generic

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

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

### Hierarchy

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

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

### HierarchBelongsTo

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

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

### Inner

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

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

### IntervalMatch

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

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

### Join

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

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

### Keep

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

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

### Left

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

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

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

### Mapping

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

```
Mapping ( 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. Для указания внешнего объединения перед явным префиксом **Join** может стоять префикс **Outer**. При внешнем объединении полученная таблица будет содержать все значения из обеих таблиц исходных данных, где значения связанных полей представлены 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 )
```

### Replace

Префикс **replace** используется для удаления таблицы в программе Qlik Sense полностью и ее замены загруженной или выбранной таблицей.

```
Replace [only] (loadstatement | selectstatement | map...usingstatement)
```

### 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** можно добавить в любой оператор **LOAD**, **SELECT** или **map...using** в скрипте. Он применяется только во время частичной загрузки.



*В настоящее время частичная перезагрузка поддерживается только при использовании Qlik Engine API.*

### Синтаксис:

```
Add [only] (loadstatement | selectstatement | mapstatement)
```

Во время частичной перезагрузки таблица Qlik Sense, для которой создается имя с помощью оператора **add LOAD/add SELECT** (если такая таблица существует), дополняется результатом выполнения оператора **add LOAD/add SELECT**. Проверка дубликатов не выполняется. Таким образом, оператор, использующий префикс **add**, будет, как правило, включать в себя квалификатор **distinct** или защитные дубликаты утверждения **where**. Оператор **map...using** запускает сопоставление данных также и во время частичного выполнения скрипта.

### Аргументы:

Аргумент	Описание
only	Дополнительный квалификатор, указывающий на то, что оператор следует игнорировать в ходе обычной (не частичной) перезагрузки.

### Примеры и результаты:

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



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

## 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:**

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. Обратите внимание, что параметр incremental может использоваться только с операторами <b>LOAD</b> и текстовыми файлами, а инкрементальная загрузка не может использоваться там, где устаревшие данные изменены или удалены!
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**.

**Пример 1:**

```
buffer SELECT * from MyTable;
```

**Пример 2:**

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

**Пример 3:**

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

**Concatenate**

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

последней созданной логической таблицей.

### Синтаксис:

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

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

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

### Аргументы:

Аргумент	Описание
tablename	Имя существующей таблицы.

### Пример:

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

## Crosstable

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

### Синтаксис:

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

### Аргументы:

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

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

### Примеры:

```
Crosstable (Month, Sales) LOAD * from ex1.csv;  
Crosstable (Month,Sales,2) LOAD * from ex2.csv;  
Crosstable (A,B) SELECT * from table3;
```

### First

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

### Синтаксис:

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

### Аргументы:

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

### Примеры:

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

### Generic

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

### Синтаксис:

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

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

### Примеры:

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

### 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 родительскими 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:

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.

Аргумент	Описание
NodeName	Имя поля, содержащего имя узла. Это поле должно существовать во входной таблице.
ParentName	Строка, которая используется для наименования нового поля <b>ParentName</b> . 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 <b>NodeName</b> .
PathName	Строка, которая используется для наименования нового поля <b>Path</b> , 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 <b>Path</b> . Дополнительный параметр. При его отсутствии используется '/
Depth	Строка, которая используется для наименования нового поля <b>Depth</b> , 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
];

```

Node ID	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:

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 <b>DepthDiff</b> , 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
];
```

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

## Inner

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

### Синтаксис:

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

### Аргументы:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> для загруженной таблицы.

### Пример 1:

Table1	
--------	--



A	B
1	aa
2	cc
3	ee

Table2	
A	C
1	xx
4	yy

QVTable:  
SQL SELECT \* From table1;  
inner join SQL SELECT \* From table2;

QVTable		
A	B	C
1	aa	xx

### Пример 2:

QVTab1:  
SQL SELECT \* From Table1;  
QVTab2:  
inner keep SQL SELECT \* From Table2;

QVTab1	
A	B
1	aa

QVTab2	
A	C
1	xx

Две таблицы в примере **keep**, разумеется, связаны посредством поля A.

## 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:

Аргумент	Описание
matchfield	Поле, содержащее дискретные числовые значения, которые нужно связать с интервалами.
keyfield	Поля, содержащие дополнительные атрибуты, сопоставляемые при преобразовании.
loadstatement or selectstatement	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 <b>IntervalMatch</b> . Интервалы всегда закрытые, т. е. конечные точки включены в интервал. Нечисловые границы приводят к тому, что интервал игнорируется (неопределенный).

### Пример 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.

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

### Пример 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:

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

### Join

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

#### Синтаксис:

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

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

#### Аргументы:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> для загруженной таблицы.

#### Пример:

```
Join LOAD * from abc.csv;
```

```
Join SELECT * from table1;
```

```
tab1:
```

```
LOAD * from file1.csv;
```

```
tab2:
```

```
LOAD * from file2.csv;
```

```
... ..
```

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

### 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:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> 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. раздел *Left* (страница 678)

### Синтаксис:

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

### Аргументы:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> для загруженной таблицы.

### Пример:

Table1	
A	B
1	aa
2	cc
3	ee

Table2	
A	C
1	xx
4	yy

QVTable:  
SELECT \* From table1;  
Left Join Sselect \* From table2;

QVTable		
A	B	C
1	aa	xx
2	cc	
3	ee	

QVTab1:  
SELECT \* From Table1;  
QVTab2:  
Left Keep SELECT \* From Table2;

QVTab1	
A	B
1	aa

2	cc
3	ee

QVTab2	
A	C
1	xx

Две таблицы в примере **keep**, разумеется, связаны посредством поля A.

```
tab1:
LOAD * From file1.csv;
tab2:
LOAD * From file2.csv;
... ..
Left Keep (tab1) LOAD * From file3.csv;
```

### Mapping

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:

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

### NoConcatenate

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

#### Синтаксис:

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

#### Пример:

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

## 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. Для указания внешнего объединения перед явным префиксом **Join** может стоять префикс **Outer**. При внешнем объединении полученная таблица будет содержать все значения из обеих таблиц исходных данных, где значения связанных полей представлены 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:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> для загруженной таблицы.

### Пример:

Table1	
A	B
1	aa
2	cc
3	ee

Table2	
A	C
1	xx
4	yy

```
SQL SELECT * from table1;
join SQL SELECT * from table2;
      ИЛИ
```

```
SQL SELECT * from table1;
outer join SQL SELECT * from table2;
```

Объединенная таблица		
A	B	C
1	aa	xx
2	cc	-
3	ee	-
4	-	yy

## Replace

Префикс **replace** используется для удаления таблицы в программе Qlik Sense полностью и ее замены загруженной или выбранной таблицей.



*В настоящее время частичная перезагрузка поддерживается только при использовании Qlik Engine API.*

### Синтаксис:

**Replace** [**only**] (loadstatement | selectstatement | map...usingstatement)

Префикс **replace** можно добавить в любой оператор **LOAD**, **SELECT** или **map...using** в скрипте. Оператор **replace LOAD/replace SELECT** отбрасывает всю таблицу Qlik Sense, для которой имя сгенерировано оператором **replace LOAD/replace SELECT**, и заменяет ее новой таблицей, содержащей результат оператора **replace LOAD/replace SELECT**. Аналогичный результат достигается во время частичной и полной перезагрузки. Оператор **replace map...using** запускает сопоставление данных также и во время частичного выполнения скрипта.

### Аргументы:

Аргумент	Описание
only	Дополнительный квалификатор, указывающий на то, что оператор следует игнорировать в ходе обычной (не частичной) перезагрузки.

### 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. раздел [Right](#) (страница 683)*

### Синтаксис:

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

### Аргументы:

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
Операторы loadstatement или selectstatement	Оператор <b>LOAD</b> или <b>SELECT</b> для загруженной таблицы.

### Примеры:

Table1	
A	B
1	aa
2	cc
3	ee

Table2	
A	C
1	xx
4	yy

QVTable:  
 SQL SELECT \* from table1;  
 right join SQL SELECT \* from table2;

QVTable		
A	B	C
1	aa	xx
4	-	yy

QVTab1:  
 SQL SELECT \* from Table1;  
 QVTab2:  
 right keep SQL SELECT \* from Table2;

QVTab1	
A	B
1	aa

QVTab2	
A	C
1	xx
4	yy

Две таблицы в примере **keep**, разумеется, связаны посредством поля A.

tab1:  
 LOAD \* from file1.csv;  
 tab2:  
 LOAD \* from file2.csv;

```
... ..  
right keep (tab1) LOAD * from file3.csv;
```

### Sample

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

#### Синтаксис:

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

#### Аргументы:

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

#### Пример:

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



Скобки допускаются, но необязательны.

### Semantic

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

#### Синтаксис:

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

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

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

#### Пример:

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

### Unless

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

#### Синтаксис:

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

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

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

#### Аргументы:

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

#### Примеры:

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

### When

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

#### Синтаксис:

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

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

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

### Синтаксис:

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

### Пример 1:

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

### Пример 2:

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

### Пример 3:

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

## 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}
```

### Binary

Оператор **binary** используется для загрузки данных из другого приложения Qlik Sense или QlikView 11.2, или более ранней версии, включая данные доступа к секции. Другие элементы приложения не включены, например, листы, истории, визуализации, основные элементы или переменные.

```
Binary file  
file ::= [ 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

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

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

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

### Derive

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

```
Derive [Field[s]] From [Field[s]] field_list Using definition
```

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

### Direct Query

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

```
Direct Query [path]
```

### Directory

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

```
Directory [path]
```

### Disconnect

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

```
Disconnect
```

### drop field

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



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

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

### drop table

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



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

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

### Execute

Оператор **Execute** используется для запуска других программ в ходе загрузки данных Qlik Sense.

Например, для выполнения необходимых преобразований.

```
Execute commandline
```

### FlushLog

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

```
FlushLog
```

### Force

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

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

### LOAD

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

```
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 ]
```

### 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 «=» 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 получают 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

Эта функция скрипта создает файл QVD или CSV.

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

### Tag

Эта функция скрипта предоставляет возможность присваивать теги одному или нескольким полям. Если делается попытка присвоить тег имени поля, отсутствующему в приложении, то эта операция будет игнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

```
Tag fields fieldlist using mapname
```

```
Tag field fieldname 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

Предоставляет возможность удалить теги из одного или нескольких полей. Если делается попытка снять тег с имени поля, отсутствующего в приложении, то эта операция будет игнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

```
Untag fields fieldlist using mapname
```

```
Untag field fieldname 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:

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 <b>SELECT</b> и <b>LOAD</b> . Новый псевдоним для имени поля может быть задан с помощью нового оператора <b>alias</b> в любой последующей точке скрипта.

### Binary

Оператор **binary** используется для загрузки данных из другого приложения Qlik Sense или QlikView 11.2, или более ранней версии, 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**, причем он должен быть первым оператором скрипта, даже перед оператором SET, который обычно расположен в начале скрипта.*

#### Синтаксис:

```
binary [path] filename
```

**Аргументы:**

Аргумент	Описание
filename	Имя файла, включая расширение файла .qvw или .qvf
path	<p>Путь к файлу, который должен быть ссылкой на подключение к данным папки. Это необходимо, если файл расположен не в рабочем каталоге Qlik Sense.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"> <li>абсолютный</li> </ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"> <li>относительно приложения, содержащего эту строку скрипта.</li> </ul> <p><b>Пример: data\</b></p>

**Примеры**

binary lib://MyData/customer.qvw;	В этом примере файл <i>customer.qvw</i> должен быть расположен в папке, подключенной к подключению данных MyData.
binary customer.qvw;	В этом примере файл <i>customer.qvw</i> 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:**

Аргумент	Описание
<i>*fieldlist</i>	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.
<i>mapname</i>	Имя таблицы сопоставления, считанной ранее в операторе сопоставления <b>LOAD</b> или <b>SELECT</b> .
<i>fieldname</i>	Имя поля, для которого необходимо добавить комментарий.
<i>comment</i>	Комментарий, который следует добавить к полю.

**Пример 1:**

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

**Пример 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:**

Аргумент	Описание
<i>tablelist</i>	(table{,table})

Аргумент	Описание
<i>mapname</i>	Имя таблицы сопоставления, считанной ранее в операторе сопоставления <b>LOAD</b> или <b>SELECT</b> .
<i>tablename</i>	Имя таблицы, для которой необходимо добавить комментарий.
<i>comment</i>	Комментарий, который следует добавить в таблицу.

**Пример 1:**

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

**Пример 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.



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



*В настоящее время подключение к базам данных OLE DB/ODBC в Qlik Sense Cloud невозможно.*

**Синтаксис:**

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

**Аргументы:**

Аргумент	Описание
connect-string	<p><code>connect-string ::= datasource { ; conn-spec-item }</code></p> <p>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.</p> <p><b>datasource</b> должен являться определенным источником данных ODBC или строкой, которая определяет поставщика OLE DB.</p> <p><code>conn-spec-item ::= DBQ=database_specifier   DriverID=driver_specifier   UID=userid   PWD=password</code></p> <p>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.</p>
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.

**Пример 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**.

**Пример 2:**

```
LIB CONNECT TO 'MyDataConnection';
```

**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 и групп, 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	<p>Имя определения поля с двоеточием в конце.</p> <div>  <p><i>Не используйте autoCalendar в качестве имени определения поля, так как это имя зарезервировано для автоматически созданных шаблонов календаря.</i></p> </div> <p><b>Пример:</b></p> <p>calendar:</p>
tag_list	<p>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.</p> <p><b>Пример:</b></p> <p>'\$date'</p>

Аргумент	Описание
parameter_list	<p>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.</p> <p><b>Пример:</b></p> <pre>first_month_of_year = 1</pre>
field_list	<p>Список полей, разделенных запятыми, которые будут созданы при использовании определения поля. Поле определяется в виде &lt;expression&gt; <b>As</b> field_name <b>tagged</b> tag. Используйте \$1 для ссылки на поле данных, из которого должны быть созданы производные поля.</p> <p><b>Пример:</b></p> <pre>Year(\$1) As Year tagged '\$year'</pre>

**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.  <b>Пример:</b>  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.  <b>Пример:</b>  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.  <b>Пример:</b>  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 данные 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:**

Аргумент	Описание
definition	Имя определения поля для использования при извлечении полей.  <b>Пример: calendar</b>
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.  <b>Пример: orderDate, shippingDate</b>
tag_list	Список тегов, разделенных запятыми. Производные поля будут созданы для всех полей данных с любым из перечисленных тегов.  <b>Пример: '\$date'</b>

**Примеры:**

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

**Direct Query**

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



*В настоящее время подключение к базам данных OLE DB/ODBC в Qlik Sense Cloud невозможно.*

**Синтаксис:**

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

```
[WHERE where_clause]
```

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

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

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



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

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

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

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

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

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

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

Примеры:

```
DIRECT QUERY
```

```
    DIMENSION Dim1, Dim2  
    MEASURE
```

```
        NATIVE ('X % Y') AS X_MOD_Y
```

```
FROM TableName  
DIRECT QUERY
```

```
    DIMENSION Dim1, Dim2
```



```
MEASURE X, Y
FROM TableName
WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```



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

### Аргументы:

Аргумент	Описание
fieldlist	Список спецификаций поля, разделенных запятыми, <i>fieldname {, fieldname}</i> . Спецификация поля может быть именем поля. В этом случае такое же имя используется для имени столбца базы данных и имени поля Qlik Sense. Также спецификация поля может быть «полем alias». В этом случае выражению базы данных или имени столбца задается имя поля Qlik Sense.
tablelist	Список имен таблиц или представлений в базе данных, из которой загружаются данные. Как правило, это представления, содержащие оператор JOIN, выполненный в базе данных.
where_ clause	<p>Здесь не приведено полное описание синтаксиса предложений базы данных <b>WHERE</b>, но большинство «реляционных выражений» SQL разрешено использовать, включая вызовы функций, оператор <b>LIKE</b> для строк, <b>IS NULL</b> и <b>IS NOT NULL</b>, а оператор <b>IN. BETWEEN</b> не включен.</p> <p><b>NOT</b> — это унарный оператор, в отличие от модификатора на определенные ключевые слова.</p> <p>Примеры:</p> <pre>WHERE x &gt; 100 AND "Region Code" IN ('south', 'west') WHERE Code IS NOT NULL and Code LIKE '%prospect' WHERE NOT x in (1,2,3)</pre> <p>Последний пример не может быть записан как:</p> <pre>WHERE x NOT in (1,2,3)</pre>

### Пример:

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

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

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



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

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

можно заменить следующим оператором **DIRECT QUERY**:

```
DIRECT QUERY
  DIMENSION
    SalesOrderID,
    RevisionNumber

  MEASURE
    SubTotal,
    TaxAmt

  DETAIL
    OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

### Списки полей Direct Discovery

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

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

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

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

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

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

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

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

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

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

### Примеры:

```
DIRECT QUERY  
  
    DIMENSION  
  
        SalesOrderID, RevisionNumber  
  
    MEASURE  
  
        SubTotal AS "Sub Total"  
  
FROM Adventureworks.Sales.SalesOrderHeader;  
  
DIRECT QUERY
```

DIMENSION

"SalesOrderID" AS "Sales Order ID"

MEASURE

SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"

FROM Adventureworks.Sales.SalesOrderHeader;

DIRECT QUERY

DIMENSION

(2\*Radius\*3.14159) AS Circumference,

Molecules/6.02e23 AS Moles

MEASURE

Num1 AS numA

FROM TableName;

DIRECT QUERY

DIMENSION

concat(region, 'code') AS region\_code

MEASURE

Num1 AS NumA

FROM TableName;

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

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

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

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

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

### Directory

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

**Синтаксис:**

```
Directory [path]
```

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

### Аргументы:

Аргумент	Описание
<b>path</b>	<p>Текст может интерпретироваться как путь к файлу qvf.</p> <p>Path — путь к файлу:</p> <ul style="list-style-type: none"><li>• абсолютный <b>Пример: c:\data\</b></li><li>• относительно рабочего каталога приложения Qlik Sense. <b>Пример: data\</b></li><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. <b>Пример: http://www.qlik.com</b></li></ul>

### Примеры:

```
Directory lib://Data/;  
Directory c:\userfiles\data;
```

## 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 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.

Пример	Результат
<pre>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;</pre>	После создания таблицы <i>Tab2</i> таблица <i>Tab1</i> удаляется.

## Execute

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



*Этот оператор не поддерживается в стандартном режиме.*



*Этот оператор не поддерживается в стандартном режиме или в Qlik Sense Cloud.*

### Синтаксис:

#### Синтаксис:

```
execute commandline
```

### Аргументы:

Аргумент	Описание
<i>commandline</i>	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
```

## 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 прописными 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
<p>В данном примере показано принудительное использование прописных букв.</p> <pre>FORCE Capitalization; Capitalization: LOAD * Inline [ ab Cd eF GH ];</pre>	<p>Таблица <b>Capitalization</b> содержит следующие значения:</p> <p>Ab Cd eF GH</p> <p>Все значения записываются прописными буквами.</p>
<p>В данном примере показано принудительное использование верхнего регистра.</p> <pre>FORCE Case Upper; CaseUpper: LOAD * Inline [ ab Cd eF GH ];</pre>	<p>Таблица <b>CaseUpper</b> содержит следующие значения:</p> <p>AB CD EF GH</p> <p>Все значения записываются в верхнем регистре.</p>

Пример	Результат
<p>В данном примере показано принудительное использование нижнего регистра.</p> <pre>FORCE Case Lower; CaseLower: LOAD * Inline [ ab Cd eF GH ];</pre>	<p>Таблица <b>CaseLower</b> содержит следующие значения:</p> <p>ab cd ef gh</p> <p>Все значения записываются в нижнем регистре.</p>
<p>В данном примере показано принудительное использование смешанного регистра.</p> <pre>FORCE Case Mixed; CaseMixed: LOAD * Inline [ ab Cd eF GH ];</pre>	<p>Таблица <b>CaseMixed</b> содержит следующие значения:</p> <p>ab Cd eF GH</p> <p>Все значения отображаются в том же виде, что и в скрипте.</p>

См. также:

## 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.


### Синтаксис:

```
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 ]
```

### Аргументы:

Аргумент	Описание
distinct	<b>distinct</b> — это логическое условие, используемое в том случае, если должна быть загружена только первая из дублирующихся записей.

Аргумент	Описание
fieldlist	<p><i>fieldlist</i> ::= ( *   <i>field</i> {, *   <i>field</i> } )</p> <p>Список полей, которые необходимо загрузить. Символ * в качестве списка полей обозначает все поля таблицы.</p> <p><i>field</i> ::= ( <i>fieldref</i>   <i>expression</i> ) [<b>as</b> <i>aliasname</i> ]</p> <p>Определение поля должно всегда содержать литерал, ссылку на существующее поле или выражение.</p> <p><i>fieldref</i> ::= ( <i>fieldname</i>   @<i>fieldnumber</i>   @<i>startpos</i>:<i>endpos</i> [ <i>I</i>   <i>U</i>   <i>R</i>   <i>B</i>   <i>T</i> ] )</p> <p><i>fieldname</i> — 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:</p> <p>@<i>fieldnumber</i> 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.</p> <p>@<i>startpos</i>:<i>endpos</i> представляет начальную и конечную позиции поля в файле с записями фиксированной длины. Позиции должны быть положительными целыми числами. Двум числами должен предшествовать символ «@», и они должны быть разделены двоеточием. Нумерация всегда начинается с 1 и содержит число позиций. В последнем поле элемент <i>n</i> используется как конечное положение.</p> <ul style="list-style-type: none"> <li>• Если после @<i>startpos</i>:<i>endpos</i> указаны символы <i>I</i> или <i>U</i>, прочитанные байты будут интерпретированы как двоичное целое число со знаком (<i>I</i>) или без знака (<i>U</i>) (порядок байтов Intel). Прочитанное число позиций должно быть 1, 2 или 4.</li> <li>• Если после @<i>startpos</i>:<i>endpos</i> указан символ <i>R</i>, 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.</li> <li>• Если после @<i>startpos</i>:<i>endpos</i> указан символ <i>B</i>, 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.</li> </ul> <p><i>expression</i> 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.</p> <p><b>as</b> используется для назначения полю нового имени.</p>

Аргумент	Описание
from	<p>Элемент <b>from</b> используется, если данные должны быть загружены из файла с помощью папки или подключения к данным из веб-файла.</p> <p><i>file ::= [ path ] filename</i></p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"> <li>абсолютный</li> </ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"> <li>относительно рабочего каталога приложения Qlik Sense.</li> </ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"> <li>URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li> </ul> <p><b>Пример: http://www.qlik.com</b></p> <p>Если путь отсутствует, программа Qlik Sense выполняет поиск файла в каталоге, указанном оператором <b>Directory</b>. Если оператора <b>Directory</b> нет, программа Qlik Sense выполняет поиск в рабочем каталоге C:\Users\{user}\Documents\Qlik\Sense\Apps.</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p> 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.</p> </div> <p>Элемент <i>filename</i> может содержать стандартные знаки подстановки DOS ( * и ? ). В результате будут загружены все файлы в указанном каталоге, удовлетворяющие критериям.</p> <p><i>format-спес ::= ( fspec-item { , fspec-item } )</i></p> <p>Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки.</p>

Аргумент	Описание
from_field	<p><b>from_field</b> используется в случае, если данные должны быть загружены из ранее загруженного поля.</p> <p><i>fieldsource::=(tablename, fieldname)</i></p> <p>Поле — это имя ранее загруженных <i>tablename</i> и <i>fieldname</i>.</p> <p><i>format-spec ::= ( fspec-item {, fspec-item } )</i></p> <p>Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки.</p>
inline	<p><b>inline</b> используется в случае, если данные должны быть введены в скрипте, а не загружены из файла.</p> <p><i>data ::= [ text ]</i></p> <p>Данные, введенные с использованием выражения <b>inline</b>, 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 выражение <b>inline</b>, например, нажав клавишу Enter при вводе в скрипте.</p> <p><i>format-spec ::= ( fspec-item {, fspec-item } )</i></p> <p>Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки.</p>
resident	<p>Элемент <b>resident</b> используется в случае, если данные должны быть загружены из ранее загруженной таблицы.</p> <p><i>table label</i> — это метка, предшествующая оператору(-ам) <b>LOAD</b> или <b>SELECT</b>, используемым для создания исходной таблицы. В конце метки должно быть указано двоеточие.</p>
autogenerate	<p><b>autogenerate</b> используется в случае, если данные должны быть автоматически созданы программой Qlik Sense.</p> <p><i>size ::= number</i></p> <p><i>Number</i> — это целое число, обозначающее число создаваемых записей.</p> <p>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 <b>Peek</b>.</p>
where	<p><b>where</b> — предложение, которое используется для указания того, нужно ли включить запись в выборку или нет. Выборка включается, если элемент <i>criterion</i> имеет значение True.</p> <p><i>criterion</i> — это логическое выражение.</p>

Аргумент	Описание
while	<p><b>while</b> — это выражение, используемое для указания необходимости повторного чтения записи. Эта же запись читается, если для элемента <i>criterion</i> указано значение True. Чтобы быть полезным, выражение <b>while</b> обычно должно содержать функцию <b>IterNo( )</b>.</p> <p><i>criterion</i> — это логическое выражение.</p>
group by	<p><b>group by</b> — 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.</p> <p><i>groupbyfieldlist ::= (fieldname {,fieldname } )</i></p>
order by	<p><b>order by</b> — это выражение, используемое для сортировки записей резидентной таблицы до их обработки оператором <b>load</b>. 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 выражение может использоваться, только если источником данных является резидентная таблица.</p> <p>Поля заказов указывают поле для сортировки резидентной таблицы. Поле может быть указано по имени или по числу в резидентной таблице (первое поле имеет номер 1).</p> <p><i>orderbyfieldlist ::= fieldname [ sortorder ] { , fieldname [ sortorder ] }</i></p> <p><i>sortorder</i> имеет значение <i>asc</i> для сортировки по возрастанию или <i>desc</i> для сортировки по убыванию. Если <i>sortorder</i> не указан, используется <i>asc</i>.</p> <p><i>fieldname</i>, <i>path</i>, <i>filename</i> и <i>aliasname</i> — это текстовые строки, представляющие подразумеваемые соответствующие имена. Любое поле в исходной таблице может использоваться в качестве <i>fieldname</i>. Однако поля, созданные с помощью выражения (<i>aliasname</i>), не рассматриваются и не могут использоваться внутри одного оператора <b>load</b>.</p>

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

#### 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 (MyData):

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

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

```
LOAD * from 'lib://MyData/*.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);
```

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 на A, а второго на 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 со 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://MyData/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 des;
```

Загрузка данных из ранее загруженных полей

Загрузка поля Types из ранее разгруженной 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 выражения **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 выражение **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

### 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 ]
```

### 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:

Аргумент	Описание
<b>utf8</b>	Набор символов UTF-8
<b>unicode</b>	Набор символов Unicode
<b>ansi</b>	Windows, кодовая страница 1252
<b>oem</b>	DOS, OS/2, AS400 и другие
<b>mac</b>	Кодовая страница 10000

Аргумент	Описание
<b>codepage is</b>	Со спецификатором <b>codepage</b> можно использовать любую кодовую страницу Windows как <i>N</i> .

**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:**

р *Load (страница 90)*

**Формат таблицы**

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

- 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 .xls/x*.
- 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:

Аргумент	Описание
char	Указывает один символ из 127 ASCII символов.

Могут использоваться следующие значения:

- 't'** представляет знак табуляции и указывается с кавычками или без них.
- '\'** представляет обратную косую черту (`\`).
- 'spaces'** представляет все комбинации одного или нескольких пробелов. Непечатаемые 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:

р *Load* (страница 90)

### 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* (страница 90)

### 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**.

### Пример 1:

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

### Пример 2:

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

---

### См. также:

p *Load* (страница 90)

### Header is

Задаёт 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 в строках (**header is line** или **header is n lines**). **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:

р *Load* (страница 90)

### 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:

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* (страница 90)

### 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:

Аргумент	Описание
no quotes	Используется, если кавычки не должны приниматься в текстовом файле.
msq	<p>Используется для задания современного стиля кавычек, которые позволяют вводить в поля многострочное содержимое. Поля, содержащие символы конца строки, необходимо заключать в двойные кавычки.</p> <p>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.</p>

### 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:**

р *Load* (страница 90)

### KML

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

**Синтаксис:**

```
kml
```

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

### 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 «=» до присваивания его переменной.

**Синтаксис:**

```
Let variablename=expression
```

Слово **let** может игнорироваться, но при этом этот оператор становится оператором управления. Такой оператор без ключевого слова **let** должен находиться в одной строке скрипта и может заканчиваться точкой с запятой или символом конца строки.

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'
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:**

Аргумент	Описание
<i>fieldlist</i>	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.
<i>mapname</i>	Имя таблицы сопоставления, считанной ранее в операторе <b>mapping load</b> или <b>mapping select</b> .

**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:**

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить <b>NullAsNull</b> . 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:

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить <b>NullAsValue</b> . 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 получают 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:

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.

### Пример 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**.

### Пример 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:

Аргумент	Описание
string	Произвольный текст.

### Пример:

```
Rem ** This is a comment **;  
/* This is also a comment */  
// This is a comment as well
```

## 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.

### Пример 1:

```
Rename Field XAZ0007 to Sales;
```

### Пример 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:

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.

### Пример 1:

```
Tab1:  
SELECT * from Trans;  
Rename Table Tab1 to Xyz;
```

### Пример 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:

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:

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.



*Этот оператор не поддерживается в Qlik Sense Cloud.*

### 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.

#### 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 соединяться 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:

Аргумент	Описание
distinct	<b>distinct</b> — это логическое условие, используемое в случае, если копии комбинаций значений в выбранных полях должны быть загружены только один раз.
	<b>distinctrow</b> — это логическое условие, используемое в случае, если копии записей в таблице источника должны быть загружены только один раз.
fieldlist	<p><b>fieldlist ::= (*  field ) {, field }</b>  Список полей, которые необходимо выбрать. Символ «*» в качестве списка полей обозначает все поля таблицы.</p> <p><b>fieldlist ::= field {, field }</b>  Список одного или нескольких полей, разделенных запятыми.</p> <p><b>field ::= ( fieldref  expression ) [as aliasname ]</b>  Выражение может, к примеру, быть числовой или строковой функцией, основанной на одном или нескольких других полях. Некоторые из обычно принимаемых операторов и функций: +, -, *, /, &amp; (объединение строк), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname) и т. д. Дополнительную информацию см. в документации к драйверу ODBC.</p> <p><b>fieldref ::= [ tablename. ] fieldname</b>  <b>tablename</b> и <b>fieldname</b> являются текстовыми строками, идентичными тому, что они подразумевают. Они должны быть заключены в прямые двойные кавычки, если они содержат, например, пробелы.  Предложение <b>as</b> используется для назначения полю нового имени.</p>
from	<p>Список таблиц, из которых выбираются поля.</p> <p>Элемент <b>tablename</b> может быть в кавычках, а может и не быть.</p>
where	<p><b>where</b> — предложение, которое используется для указания того, нужно ли включить запись в выборку или нет.</p> <p><b>criterion</b> является логическим выражением, которое иногда может быть очень сложным. Некоторые из принимаемых операторов: числовые операторы и функции, =, &lt;&gt; или # (не равно), &gt;, &gt;=, &lt;, &lt;=, <b>and</b>, <b>or</b>, <b>not</b>, <b>exists</b>, <b>some</b>, <b>all</b>, <b>in</b>, а также новые операторы <b>SELECT</b>. Дополнительную информацию можно получить в документации драйвера ODBC или поставщика OLE DB.</p>
group by	<p><b>group by</b> — 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.</p>

Аргумент	Описание
having	<b>having</b> — это предложение, используемое для классификации групп подобно тому, как предложение <b>where</b> используется для классификации записей.
order by	<b>order by</b> — предложение, используемое для указания порядка сортировки результирующей таблицы оператора <b>SELECT</b> .
join	<b>join</b> — это префикс, 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 <b>Connect</b> , определит используемые кавычки.

**Пример 1:**

```
SELECT * FROM `Categories`;
```

**Пример 2:**

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

**Пример 3:**

```
SELECT `Order ID`, `Product ID`,  
`Unit Price` * Quantity * (1-Discount) as NetSales  
FROM `Order Details`;
```

**Пример 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
```

**Пример 1:**

```
Set FileToUse=Data1.csv;
```

**Пример 2:**

```
Set Constant="My string";
```

### Пример 3:

```
Set BudgetYear=2012;
```

### Sleep

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

#### Синтаксис:

```
Sleep n
```

#### Аргументы:

Аргумент	Описание
n	Задается в миллисекундах, где <i>n</i> — положительное целое число, не превышающее 3600000 (то есть 1 час). В качестве значения может выступать выражение.

### Пример 1:

```
Sleep 10000;
```

### Пример 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
<i>sql_command</i>	Допустимая команда SQL.

### Пример 1:

```
SQL Leave;
```

### Пример 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:

Аргумент	Описание
string	<p>Произвольный текст. Обратите внимание, что при наличии в строке пробелов она должна быть заключена в кавычки.</p> <p>Если значение не указано, то по умолчанию используется <b>star is</b>; то есть символ звездочки отсутствует, если он не будет указан явным образом. Это действительно до тех пор, пока не будет создан новый оператор <b>star</b>.</p>

### Пример:

Следующий пример представляет собой извлечение из скрипта загрузки данных, описывающее доступ к секции.

```
Star is *;
```

```
Section Access;  
LOAD * INLINE [  
ACCESS, USERID, PASSWORD, OMIT  
ADMIN, ADMIN, ADMIN,  
USER, USER1, U1, SALES  
USER, USER2, U2, WAREHOUSE  
USER, USER3, U3, EMPLOYEES  
USER, USER4, U4, SALES  
USER, USER4, U4, WAREHOUSE  
USER, USER5, U5, *  
];
```

```
Section Application;  
LOAD * INLINE [  
SALES, WAREHOUSE, EMPLOYEES, ORDERS  
1, 2, 3, 4  
];
```

Применяются следующие условия:

- *Star* соответствует символу *\**.
- 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*.
- Для пользователя *USER5* добавлен символ «*\**», который означает, что все поля в списке *OMIT* недоступны. Символ звездочки *\** соответствует всем перечисленным значениям, а не всем значениям поля.



- Пользователь *USER5* не видит поля *SALES*, *WAREHOUSE* и *EMPLOYEES*, однако видит поле *ORDERS*.

## Store

Эта функция скрипта создает файл QVD или CSV.

### Синтаксис:

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

Оператор создаст файл QVD или CSV 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:

Аргумент	Описание
<i>fieldlist</i> ::= ( *   <i>field</i> ) { , <i>field</i> }	<p>Список полей, которые необходимо выбрать. Символ «*» в качестве списка полей обозначает все поля.</p> <p><i>field</i>::= <i>fieldname</i> [ <b>as</b> <i>aliasname</i> ]</p> <p><i>fieldname</i> — это текст, идентичный имени поля в элементе <i>table</i>. (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.)</p> <p><i>aliasname</i> — альтернативное имя поля, которое предназначено для использования в результирующем файле QVD или CSV.</p>
<i>table</i>	Метка скрипта, представляющая уже загруженную таблицу, которую планируется использовать в качестве источника данных.

Аргумент	Описание
<i>filename</i>	<p>Имя целевого файла, включая действительный путь к существующему подключению к данным папки.</p> <p><b>Пример: 'lib://Table Files/target.qvd'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"> <li>абсолютный</li> </ul> <p><b>Пример: c:\data\sales.qvd</b></p> <ul style="list-style-type: none"> <li>относительно рабочего каталога приложения Qlik Sense.</li> </ul> <p><b>Пример: data\sales.qvd</b></p> <p>Если путь отсутствует, программа Qlik Sense сохраняет файл в каталоге, указанном оператором <b>Directory</b>. Если оператора <b>Directory</b> нет, программа Qlik Sense сохраняет файл в рабочем каталоге C:\Users\{user}\Documents\Qlik\Sense\Apps.</p>
<i>format-spec ::= ( <b>txt</b>   <b>qvd</b> )</i>	<p>С целью указания формата используется текст <b>txt</b> для обозначения текстовых файлов или текст <b>qvd</b> — для файлов qvd. Если формат не указан, то используется <b>qvd</b>.</p>

**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';
```

**Tag**

Эта функция скрипта предоставляет возможность присваивать теги одному или нескольким полям. Если делается попытка присвоить тег имени поля, отсутствующему в приложении, то эта операция будет игнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

**Синтаксис:**

```
Tag fields fieldlist using mapname
Tag field fieldname with tagname
```

### Аргументы:

Аргумент	Описание
fieldlist	Разделенный запятыми список полей, которые следует разметить, начиная с этой точки выполнения скрипта.
mapname	Имя таблицы сопоставления, считанной ранее в операторе <b>mapping Load</b> или <b>mapping Select</b> .
fieldname	Имя поля, для которого необходимо добавить тег.
tagname	Имя тега, применяемого к полю.

### Пример 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

### Пример 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
```

### Пример 1:

```
Trace Main table loaded;
```

### Пример 2:

```
Let MyMessage = NoOfRows('MainTable') & ' rows in Main Table';
Trace $(MyMessage);
```

## Unmap

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

**Синтаксис:****Unmap** \*fieldlist**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:**

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 <b>Qualify</b> .

**Пример 1:**

Unqualify \*;

**Пример 2:**

Unqualify TransID;

### Untag

Предоставляет возможность удалить теги из одного или нескольких полей. Если делается попытка снять тег с имени поля, отсутствующего в приложении, то эта операция будет игнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

#### Синтаксис:

```
Untag fields fieldlist using mapname
Untag field fieldname with tagname
```

#### Аргументы:

Аргумент	Описание
fieldlist	Список полей, разделенных запятыми, теги которых следует удалить.
mapname	Имя таблицы сопоставления, загруженной ранее в оператор сопоставления <b>LOAD</b> или <b>SELECT</b> .
fieldname	Имя поля, с которого необходимо снять тег.
tagname	Имя тега, который следует снять с поля.

#### Пример 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

#### Пример 2:

```
Untag field Alpha with MyTag2;
```

### 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.

## 2.4 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 определяются с помощью обозревателя переменных или в скрипте с помощью редактора загрузки данных, переменная получает свое значение от оператора **Let**, **Set** или от других операторов управления в скрипте загрузки данных.



*При редактировании листа можно также работать с переменными Qlik Sense с помощью окна обзора переменных.*

### 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.

### Определение переменной

Ниже представлен синтаксис для определения переменной:

```
set variablename = string
```

или

```
let variable = expression
```

используется. Команда **Set** присваивает текст справа от знака равенства переменной, в то время как команда **Let** вычисляет выражение.

В переменных учитывается регистр.



*Не рекомендуется использовать одинаковые имена для переменной и поля или функции в Qlik Sense.*

### Примеры:

`set hidePrefix = $ ; //` в переменной символ «\$» будет получен как значение.

`let vToday = Num(Today()); //` возвращает серийный номер сегодняшней даты.

## 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 окна обзора переменных.

## 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
```

## 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 неизменной 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:

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** *страница 148*

**Переменные интерпретации числа** *страница 136*

**Системные переменные** *страница 128*

**Значение, обрабатывающее переменные** *страница 134*

## 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 дисководов 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 дискковод 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. 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.*



*В среде распределенного узла файлы индекса поиска создаются для каждого узла отдельно, распределения по узлам не происходит.*

### Пример 1: Создавать файлы индекса поиска в ходе повторной загрузки данных

```
set CreateSearchIndexOnReload=1;
```

### Пример 2: Создавать поля индекса поиска после первого поискового запроса

```
set CreateSearchIndexOnReload=0;
```



*Действительно только для синхронизированной устойчивости: в многоузловой среде, если узел перезагрузки и узел выполнения поиска не совпадают, необходимо отключить создание индекса поиска в ходе перезагрузки. Если данный параметр не отключен, в ходе перезагрузки будет создан индекс, что приводит к ненужным затратам времени и пространства на диске.*

## 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. 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://.



*Конструкция **set Include =filename** не применяется.*

#### Примеры:

```
$(Include=abc.txt);
$(Must_Include=lib://MyDataFiles\abc.txt);
```

### 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;
```

## Значение, обрабатывающее переменные

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.

## Переменные интерпретации числа

Переменные интерпретации числа определяет система, то есть они создаются автоматически при создании нового приложения в соответствии с текущими региональными настройками операционной системы. В Qlik Sense Desktop это выполняется согласно настройкам операционной системы компьютера, а в Qlik Sense это выполняется согласно операционной системе сервера, на котором установлена программа Qlik Sense.

Переменные указываются в верхней части скрипта нового приложения Qlik Sense и заменяют стандартные настройки операционной системы для определенных параметров формата чисел во время выполнения скрипта. Эти переменные можно свободно удалять, редактировать или копировать.





*Если необходимо создать приложение для определенной локали, самый простой способ, возможно, это использование Qlik Sense Desktop на компьютере с требуемыми параметрами локали в операционной системе для создания приложения. В таком случае приложение будет содержать соответствующие региональные параметры этой локали, и вы можете переместить ее на сервер Qlik Sense для дальнейших разработок.*

### Обзор переменных интерпретации числа

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

#### Форматирование валюты

##### **MoneyDecimalSep**

Указанный десятичный разделитель заменяет символ десятичного знака для валюты, используемый в операционной системе (региональные настройки).

**MoneyDecimalSep**

##### **MoneyFormat**

Указанный символ заменяет символ валюты, используемый в операционной системе (региональные настройки).

**MoneyFormat**

##### **MoneyThousandSep**

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

**MoneyThousandSep**

#### Формат чисел

##### **DecimalSep**

Заданный десятичный разделитель заменяет символ десятичного знака, используемый в операционной системе (региональные настройки).

**DecimalSep**

##### **ThousandSep**

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

**ThousandSep**

### Форматирование времени

#### **DateFormat**

Указанный формат заменяет формат даты, используемый в операционной системе (региональные настройки).

**DateFormat**

#### **TimeFormat**

Указанный формат заменяет формат времени, используемый в операционной системе (региональные настройки).

**TimeFormat**

#### **TimestampFormat**

Указанный формат заменяет форматы даты и времени, используемые в операционной системе (региональные настройки).

**TimestampFormat**

#### **MonthNames**

Указанный формат заменяет обозначение имен месяцев, используемое в операционной системе (региональные настройки).

**MonthNames**

#### **LongMonthNames**

Указанный формат заменяет обозначение полных имен месяцев, используемое в операционной системе (региональные настройки).

**LongMonthNames**

#### **DayNames**

Указанный формат заменяет имена дней недели, используемые в операционной системе (региональные настройки).

**DayNames**

#### **LongDayNames**

Указанный формат заменяет обозначение полных имен дней недели, используемое в операционной системе (региональные настройки).

**LongDayNames**

#### **FirstWeekDay**

Целое число, которое определяет, какой день использовать в качестве первого дня недели.

**FirstWeekDay**

### BrokenWeeks

этот параметр определяет, какими должны быть недели: целыми или разбитыми.

#### **BrokenWeeks**

### ReferenceDay

Этот параметр определяет, какой день в январе должен быть задан в качестве дня ссылки, чтобы определить неделю 1.

#### **ReferenceDay**

### FirstMonthOfYear

С помощью этой настройки определяется месяц, который будет использован в качестве первого месяца года. Его можно использовать для определения финансовых годов, в которых используется смещение по месяцам, например, начало будет 1 апреля.



*Данная настройка в настоящее время не используется, но зарезервирована для будущего использования.*

Допустимые настройки: от 1 (январь) до 12 (декабрь). Параметр по умолчанию — 1.

#### **Синтаксис:**

#### **FirstMonthOfYear**

#### **Пример:**

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

### BrokenWeeks

этот параметр определяет, какими должны быть недели: целыми или разбитыми.

#### **Синтаксис:**

#### **BrokenWeeks**

По умолчанию в функциях Qlik Sense используются целые недели. Это означает следующее:

- В одних годах 1-я неделя начинается в декабре, а в других годах 52-я или 53-я неделя заканчивается в январе.
- В 1-ой неделе всегда не менее четырех дней в январе.

В качестве альтернативы можно использовать разбиение недель.

- 52-я или 53-я неделя не будет продолжена в январе следующего года.
- 1-я неделя будет начинаться 1 января и в большинстве случаев она будет неполной.

Могут использоваться следующие значения:

- 0 (= использовать целые недели)
- 1 (= использовать разбитые недели)

### Примеры:

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

## DateFormat

Указанный формат заменяет формат даты, используемый в операционной системе (региональные настройки).

### Синтаксис:

**DateFormat**

### Примеры:

```
Set DateFormat='M/D/YY'; //(US format)
Set DateFormat='DD/MM/YY'; //(UK date format)
Set DateFormat='YYYY-MM-DD'; //(ISO date format)
```

## DayNames

Указанный формат заменяет имена дней недели, используемые в операционной системе (региональные настройки).

### Синтаксис:

**DayNames**

### Пример:

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

## DecimalSep

Заданный десятичный разделитель заменяет символ десятичного знака, используемый в операционной системе (региональные настройки).

### Синтаксис:

**DecimalSep**

### Примеры:

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

## FirstWeekDay

Целое число, которое определяет, какой день использовать в качестве первого дня недели.

### Синтаксис:

#### **FirstWeekDay**

В функциях Qlik Sense понедельник является первым днем недели по умолчанию. Могут использоваться следующие значения:

- 0 (= понедельник)
- 1 (= вторник)
- 2 (= среда)
- 3 (= четверг)
- 4 (= пятница)
- 5 (= суббота)
- 6 (= воскресенье)

### Примеры:

```
Set FirstWeekDay=6; //(set Sunday as the first day of the week)
```

## LongDayNames

Указанный формат заменяет обозначение полных имен дней недели, используемое в операционной системе (региональные настройки).

### Синтаксис:

#### **LongDayNames**

### Пример:

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

## LongMonthNames

Указанный формат заменяет обозначение полных имен месяцев, используемое в операционной системе (региональные настройки).

### Синтаксис:

#### **LongMonthNames**

### Пример:

```
Set LongMonthNames='January;February;March;April;May;June - -
```

## MoneyDecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака для валюты, используемый в операционной системе (региональные настройки).

### Синтаксис:

#### **MoneyDecimalSep**

### Пример:

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

### MoneyFormat

Указанный символ заменяет символ валюты, используемый в операционной системе (региональные настройки).

#### Синтаксис:

```
MoneyFormat
```

### Пример:

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

### MoneyThousandSep

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

#### Синтаксис:

```
MoneyThousandSep
```

### Пример:

```
Set MoneyThousandSep=', ';
```

### MonthNames

Указанный формат заменяет обозначение имен месяцев, используемое в операционной системе (региональные настройки).

#### Синтаксис:

```
MonthNames
```

### Пример:

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

### ReferenceDay

Этот параметр определяет, какой день в январе должен быть задан в качестве дня ссылки, чтобы определить неделю 1.

#### Синтаксис:

```
ReferenceDay
```

По умолчанию в функциях Qlik Sense используется 4 как день ссылки. Это значит, что неделя 1 должна содержать значение «январь 4», или, другими словами, в неделе 1 всегда должно быть не меньше 4 дней в январе.

Используйте следующие значения, чтобы задать день ссылки:

- 1 (= январь 1)
- 2 (= январь 2)
- 3 (= январь 3)
- 4 (= январь 4)
- 5 (= январь 5)
- 6 (= январь 6)
- 7 (= январь 7)

### Примеры:

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

### ThousandSep

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

#### Синтаксис:

```
ThousandSep
```

### Примеры:

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

### TimeFormat

Указанный формат заменяет формат времени, используемый в операционной системе (региональные настройки).

#### Синтаксис:

```
TimeFormat
```

### Пример:

```
Set TimeFormat='hh:mm:ss';
```

### TimestampFormat

Указанный формат заменяет форматы даты и времени, используемые в операционной системе (региональные настройки).

#### Синтаксис:

```
TimestampFormat
```

### Пример:

```
Set TimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

## 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;';
```

## Символьные переменные Direct Discovery

### 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, используемым 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]';
```

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

### 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:

Аргумент	Описание
<b>ErrorMode=1</b>	Настройка по умолчанию. Выполнение скрипта останавливается, и пользователь получает запрос на выполнение действия (в непакетном режиме).
<b>ErrorMode=0</b>	Программа Qlik Sense просто проигнорирует ошибку и продолжит выполнение скрипта со следующего оператора скрипта.
<b>ErrorMode=2</b>	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:

Код ошибки	Описание
0	Нет ошибки
1	Общая ошибка
2	Ошибка синтаксиса
3	Общая ошибка ODBC
4	Общая ошибка OLE DB

Код ошибки	Описание
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
13	Ошибка BIFF
14	Зашифрованная ошибка BIFF
15	Ошибка BIFF неподдерживаемой версии
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
```

## 2.5 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:

expression ::= (constant	constant	
	fieldref	
	operator1 expression	
	expression operator2 expression	
	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 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 визуализации

Выражение — это комбинация функций, полей и математических операторов (+ \* / =). 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 это можно достичь с помощью префикса TOTAL, анализа множеств или их комбинации.

Способ	Описание
Префикс TOTAL	<p>Использование префикса total в функции агрегирования игнорирует значение измерения.</p> <p>Агрегирование будет выполнено в отношении всех возможных значений поля.</p> <p>После префикса <b>TOTAL</b> 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.</p>

Способ	Описание
Анализ множеств	Использование анализа множеств в агрегировании переопределяет выборку. Агрегирование будет выполнено в отношении всех значений по всем измерениям.
Префикс TOTAL и анализ множеств	Использование префикса <b>TOTAL</b> и анализа множеств в агрегировании переопределяет выборку и игнорирует измерения.
Префикс ALL	Использование префикса <b>ALL</b> в агрегировании игнорирует выборку и измерения. Можно получить эквивалент с оператором анализа множеств {1} и префиксом <b>TOTAL</b> :  =sum(All Sales)  =sum({1} Total Sales)

#### Пример: Префикс TOTAL

В следующем примере показано, как префикс TOTAL может быть использован для вычисления доли совместного использования. При условии, что выбран элемент Q2, при использовании префикса TOTAL рассчитывается сумма всех значений без учета измерений.

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 **Number formatting** параметр **Number**, а в разделе **Formatting** выберите параметр **Simple** 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.

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
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: Префикс TOTAL и анализ множеств

В следующем примере показано, как анализ множеств и префикс 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.

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 Синтаксис для множеств

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_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 Модификаторы множества

Множество может быть изменено дополнительной или измененной выборкой. Подобное изменение может быть записано в выражении множества.

Модификатор состоит из одного или нескольких имен полей, за каждым из которых следует выборка, которая должна быть составлена на основе поля и заключена в угловые скобки: < >. Например, элемент <Year={2007,2008},region={US}>. Имена полей и значения полей можно заключить в кавычки как обычно, например: <[Sales region]={'west coast', 'South America'}>.

Установленный модификатор изменяет выборку предыдущего установленного идентификатора. Если отсутствует ссылка на установленный идентификатор, состояние текущей выборки является неявным.

Определить выборку можно несколькими способами.

- На основе другого поля
- На основе множеств элементов (список значений поля в модификаторе)
- Принудительное исключение

Данные способы описаны в следующих подразделах.

#### На основе другого поля

Простым случаем является выборка на основе выбранных значений другого поля, например: <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.

#### На основе множеств элементов (список значений поля в модификаторе)

Наиболее распространенным случаем является выборка, основанная на списке значений поля, заключенном в фигурные скобки, значения разделены запятыми. Например: <Year = {2007, 2008}>. Здесь фигурные скобки определяют множество элементов, в котором элементы могут быть

значениями поля или поисками значений поля. Поиск всегда определяется использованием двойных кавычек. Например, элемент `<Ingredient = {"*garlic*"}` выберет все ингредиенты, включая строку «чеснок». В поиске учитывается регистр, поиск выполняется для всех исключенных значений.

## Принудительное исключение

Наконец, для полей в режиме логического «AND» существует также возможность принудительного исключения. При необходимости принудительно исключить определенные значения поля потребуется использовать знак «~» (тильда) перед именем поля.



Режим логического «AND» поддерживается только при использовании API-интерфейса подсистемы Qlik.

### Примеры и результаты:

Примеры	Результаты
<code>sum( {1&lt;Region= {USA} &gt;} Sales )</code>	Возвращает продажи в регионе USA, игнорируя текущую выборку
<code>sum( {\$&lt;Region = &gt;} Sales )</code>	Возвращает продажи для текущей выборки, но выборка в элементе «Region» удаляется.
<code>sum( &lt;Region = &gt; } Sales )</code>	Возвращает то же, что и в примере выше. Если множество для изменения отсутствует, используется знак \$.
	Синтаксис в двух предыдущих примерах интерпретируется как «выборки отсутствуют» в поле «Region», т. е. будут возможны все регионы, которым присвоены другие выборки. Не эквивалентен синтаксису <code>&lt;Region = {}</code> (или любому другому тексту с правой стороны от знака равенства, неявно возникшему в результате пустого множества элементов), интерпретируемому как «регион отсутствует».
<code>sum( {\$&lt;Year = {2000}, Region = {US, SE, DE, UK, FR}&gt;} Sales )</code>	Возвращает продажи для текущей выборки, но с новыми выборками в элементах «Year» и «Region».
<code>sum( {\$&lt;~Ingredient = {"*garlic*"}} Sales )</code>	<p>Поле <i>Ingredient</i> находится в режиме логич. «AND».</p> <p>Возвращает продажи для текущей выборки, но с принудительным исключением всех ингредиентов, содержащих строку «garlic».</p>
<code>sum( {\$&lt;Year = {"2*"}&gt;} Sales )</code>	Возвращает продажи для текущей выборки, но все года начинаются на цифру «2», т. е. в поле «Year» выбран год 2000 и далее.

Примеры	Результаты
<code>sum( {\$&lt;Year = {"2*", "198*"}&gt;} Sales )</code>	Как и выше, но также 1980-е годы включены в выборку.
<code>sum( {\$&lt;Year = {"&gt;1978&lt;2004"}&gt;} Sales )</code>	Возвращаются значения продаж для текущих выборок, но с числовым поиском, используемым для оценки диапазона лет для суммирования продаж за это время.

## Модификаторы множества с операторами множества

Выборка в поле может быть определена с помощью операторов множества при работе с различными множествами элементов. Например, модификатор `<Year = {"20*", 1997} - {2000}>` выберет все года, начиная с «20» в дополнение к «1997», кроме «2000».

### Примеры и результаты:

Примеры	Результаты
<code>sum( {\$&lt;Product = Product + {OurProduct1} – {OurProduct2}&gt;} Sales )</code>	Возвращает продажи для текущей выборки, но в список выбранных продуктов добавляется продукт «OurProduct1» и удаляется продукт «OurProduct2».
<code>sum( {\$&lt;Year = Year + {"20*", 1997} – {2000}&gt;} Sales )</code>	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.
<code>sum( {\$&lt;Year = (Year + {"20*", 1997}) – {2000}&gt;} Sales )</code>	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.
<code>sum( {\$&lt;Year = {"*"} – {2000}, Product = {"*bearing*"}&gt;} Sales )</code>	Возвращает продажи для текущей выборки, но с новой выборкой в поле «Year»: все года, кроме 2000; и только для продуктов, содержащих строку «произведение».

## Модификаторы множества, использующие назначения с операторами множества implicit

Эта нотация определяет новые выборки, игнорируя текущие выборки в поле. Однако, если требуется основать выборку на текущей выборке в поле и добавить значения поля, например, необходим модификатор `<Year = Year + {2007, 2008}>`. Простой и эквивалентный способ записать это — `<Year += {2007, 2008}>`, т. е. оператор назначения неявно определяет объединение. Также неявные пересечения, исключения и симметрические разности могут быть определены с помощью элементов `"*"`, `"-"` и `"/"`.

### Примеры и результаты:

Примеры	Результаты
<code>sum( {\$&lt;Product += {OurProduct1, OurProduct2} &gt;} Sales )</code>	Возвращает продажи для текущей выборки, но с использованием неявного объединения для добавления продуктов «OurProduct1» и «OurProduct2» в список выбранных продуктов.
<code>sum( {\$&lt;Year += {"20*", 1997} – {2000} &gt;} Sales )</code>	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 <code>&lt;Year=Year + ({"20*", 1997}–{2000})&gt;</code> .
<code>sum( {\$&lt;Product *= {OurProduct1} &gt;} Sales )</code>	Возвращает продажи для текущей выборки, но только для пересечения выбранных на данный момент продуктов и продукта «OurProduct1».

## Модификаторы множества с расширенным поиском

Для определения множеств может использоваться расширенный поиск с помощью подстановочных знаков и агрегирований.

### Примеры и результаты:

Примеры	Результаты
<code>sum( {\$–1&lt;Product = {"*Internal*", "*Domestic*"}&gt;} Sales )</code>	Возвращает продажи для текущей выборки за исключением транзакций, относящихся к продуктам со строкой «Internal» или «Domestic» в имени продукта.
<code>sum( {\$&lt;Customer = {"=Sum ({1&lt;Year = {2007}&gt;} Sales ) &gt; 1000000"}&gt;} Sales )</code>	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

В выражениях множества могут использоваться переменные и другие расширения со знаком доллара.

### Примеры и результаты:

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

## Модификаторы множества с определениями значений поля `implicit`

Далее описано, как определить множество значений поля с помощью вложенного определения множества.

В подобных случаях должны использоваться функции элементов `P()` и `E()`, представляющие множество элементов возможных значений и исключенные значения поля, соответственно. В скобках можно указать одно выражение множества и одно поле. Например: `P({1} Customer)`. Эти функции не могут использоваться в других выражениях.



*Функции элементов `P()` и `E()` можно использовать только с натуральным набором. Это набор записей, которые можно определить путем простой выборки. Например, набор, заданный значением `{1-$}`, не всегда можно определить путем выборки, следовательно, он не является натуральным набором. При использовании этих функций с наборами, не являющимися натуральными, результаты могут быть неудовлетворительными.*



**Примеры и результаты:**

Примеры	Результаты
<pre>sum( {\$&lt;Customer = P ({1&lt;Product= {'Shoe'}}&gt;} Customer)&gt;} Sales )</pre>	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.
<pre>sum( {\$&lt;Customer = P ({1&lt;Product= {'Shoe'}}&gt;})&gt;} Sales )</pre>	Так же, как выше. Если в функции элемента поле опущено, функция вернет возможные значения для поля, указанного во внешнем назначении.
<pre>sum( {\$&lt;Customer = P ({1&lt;Product= {'Shoe'}}&gt;} Supplier)&gt;} Sales )</pre>	Возвращает продажи для текущей выборки, но только для тех клиентов, которые когда-либо поставляли продукт «Shoe». Здесь функция элемента P( ) возвращает список возможных поставщиков, подразумеваемых выборкой «Shoe» в поле Product. Список поставщиков затем используется в качестве выборки в поле Customer.
<pre>sum( {\$&lt;Customer = E ({1&lt;Product= {'Shoe'}}&gt;})&gt;} Sales )</pre>	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.

### 3.4 Выражение визуализации и синтаксис агрегирования

Синтаксис, используемый для выражений визуализации (диаграммы) и агрегирований, описан в следующих разделах.

#### Общий синтаксис выражений диаграммы

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 ::= aggexpression { , aggexpression }

Число и типы параметров не являются произвольными. Они зависят от используемой функции.

### Общий синтаксис для агрегирования

aggexpression ::= ( fieldref	
operator1 aggexpression	
aggexpression operator2 aggexpression	
functioninaggr	
( aggexpression )	)

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.

<b>bitnot</b>	Побитовое отрицание.	Унарный оператор. Операция применяет логическое отрицание к каждому биту операнда.
---------------	----------------------	--

**Пример:**

Элемент `bitnot 17` возвращает -18

<b>bitand</b>	Побитовое И.	Операция применяет логическое И к каждому биту операндов.
---------------	--------------	---

**Пример:**

Элемент `17 bitand 7` возвращает 1

<b>bitor</b>	Побитовое ИЛИ.	Операция применяет логическое ИЛИ к каждому биту операндов.
--------------	----------------	---

**Пример:**

Элемент `17 bitor 7` возвращает 23

<b>bitxor</b>	Побитовое исключающее ИЛИ.	Операция применяет логическое исключающее ИЛИ к каждому биту операндов.  <b>Пример:</b>  Элемент 17 bitxor 7 возвращает 22
<b>&gt;&gt;</b>	Битовый сдвиг вправо.	Операция возвращает первый операнд, сдвинутый вправо. Количество шагов определяется во втором операнде.  <b>Пример:</b>  Элемент 8 >> 2 возвращает 2
<b>&lt;&lt;</b>	Битовый сдвиг влево.	Операция возвращает первый операнд, сдвинутый влево. Количество шагов определяется во втором операнде.  <b>Пример:</b>  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).

<b>not</b>	Логическое отрицание. Один из нескольких унарных операторов. Операция возвращает логическое отрицание операнда.
<b>and</b>	Логическое И. Операция применяет логическое И к операндам.
<b>or</b>	Логическое ИЛИ. Операция возвращает логическое ИЛИ операндов.
<b>Xor</b>	Логическое исключающее ИЛИ. Операция возвращает результат операции логического исключающего 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.

<b>+</b>	Знак положительного числа (унарный оператор) или арифметического сложения. Бинарная операция возвращает сумму двух операндов.
<b>-</b>	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.

<	Меньше, чем	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, чем	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.
<>	Не равно	Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
<b>precedes</b>		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' возвращаетFALSE

при этом

' 1' precedes ' 2' возвращаетTRUE

в качестве значения пробела ASCII (' '), который имеет

меньшее значение, чем значение числа ASCII.

Сравните с:

'1' < '2' возвращает TRUE

и

' 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

при этом

'2' follows '1' возвращает TRUE

в качестве значения пробела ASCII (' '), который имеет меньшее значение, чем значение числа ASCII.

Сравните с:

' 2' > ' 1' возвращает TRUE

и

' 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 в скриптах и выражениях диаграммы

В данном разделе описаны функции, которые можно использовать в скриптах загрузки данных Qlik Sense и выражениях диаграмм для преобразования и агрегирования данных.

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 Функции агрегирования

Семейство функций, известных как функции агрегирования, состоит из функций, для которых несколько значений поля являются вводимым значением и которые возвращают один результат. В данных функциях агрегирование определяется измерением диаграммы или предложением **group by** в скрипте. 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.

#### Использование функций агрегирования в скрипте загрузки данных

Функции агрегирования могут использоваться только с помощью операторов **LOAD**.

#### Использование функций агрегирования в выражениях диаграмм

Выражение аргумента одной функции агрегирования не должно содержать другую функцию агрегирования.

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Функция агрегирования агрегирует набор возможных записей, определенных выборкой. Однако альтернативное множество записей может быть определено выражением множества в анализе множеств.



### 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 расширенных агрегирований, в которых функция **Aggr** заключена в другую функцию агрегирования с помощью диапазона результатов, полученных из функции **Aggr** в качестве ввода в агрегирование, в которое она вложена.

#### Синтаксис:

```
Aggr ({SetExpression} [DISTINCT] [NODISTINCT] expr, StructuredParameter{, StructuredParameter})
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
expr	Выражение, состоящее из функции агрегирования. По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой.
StructuredParameter	<p>StructuredParameter представляет собой измерение, к которому в некоторых случаях добавляется критерий сортировки следующего формата: (Dimension (Sort-type, Ordering))</p> <p>Измерение представляет собой одиночное поле, оно не может быть выражением. Измерение предназначено для определения диапазона значений, для которых вычисляется выражение Aggr.</p> <p>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.</p> <p>Сведения о порядке использования критериев сортировки см. в разделе <a href="#">Добавление критериев сортировки к измерению в составе структурированного параметра</a>.</p>
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 префикс <b>distinct</b> , 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.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
NODISTINCT	Если перед аргументом выражения стоит префикс <b>nodistinct</b> , то каждая комбинация значений измерений может создавать несколько возвращаемых значений в зависимости от базовой структуры данных. Если измерение только одно, функция <b>aggr</b> 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 продаж клиентом путем сложения сумм продаж 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:

## 5 Функции в скриптах и выражениях диаграммы

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 и результаты:

Пример	Результат
Avg(Aggr(Sum (UnitSales*UnitPrice), Customer))	<p>Выражение Aggr(Sum(UnitSales*UnitPrice), Customer) вычисляет общее значение продаж для значения <b>Customer</b> и возвращает несколько значений: 295, 715 и 120 для трех значений <b>Customer</b>.</p> <p>По сути, мы построили временный список значений, не создавая отдельную таблицу или столбец с этими значениями.</p> <p>Данные значения выполняют функцию вводимых данных для функции <b>Avg()</b>, служащей для вычисления среднего значения продаж, 376,6667. (Необходимо выбрать параметр <b>Итоги</b> в пункте <b>Представление</b> на панели свойств).</p>
Aggr(NODISTINCT Max (UnitPrice), Customer)	<p>Диапазон значений: 16, 16, 16, 25, 25, 25, 19 и 19. Префикс <b>nodistinct</b> означает, что диапазон содержит по одному элементу для каждой строки в данных источника: каждый является максимальным значением <b>UnitPrice</b> для каждого элемента <b>Customer</b> и <b>Product</b>.</p>

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
<code>max(aggr(sum(Customers)-above(Sum(Customers)), (MonthYear, (NUMERIC, ASCENDING))))</code>	<p>Использование критериев сортировки в аргументе <code>StructuredParameter</code>, входящем в состав выражения:</p> <p><code>max(aggr(sum(Customers)-above(Sum(Customers)), (MonthYear, (NUMERIC, ASCENDING))))</code></p> <p>В случае отсутствия критериев сортировки результат выражения <code>max(aggr(sum(Customers)-above(Sum(Customers)), (MonthYear))</code> зависит от порядка сортировки измерения <code>MonthYear</code>. Полученный результат может быть неудовлетворительным.</p> <p>За счет добавления к измерению значений типа сортировки и типа упорядочивания к структурированному параметру добавляется критерий сортировки: <code>(MonthYear, (NUMERIC, ASCENDING))</code>, где тип сортировки <code>NUMERIC</code> и тип упорядочивания <code>ASCENDING</code> указывают на сортировку <code>MonthYear</code> в числовом порядке по возрастанию.</p> <p>Рассмотрим случай, когда необходимо вычислить значение самого высокого прироста количества клиентов по сравнению с предыдущим месяцем. Это значением можно использовать, например, в визуализации ключевого показателя эффективности.</p> <p>Часть выражения <code>Aggr</code> сравнивает общее количество клиентов за месяц (заданное параметром <code>MonthYear</code>) с общим количеством клиентов за предыдущий месяц.</p> <p>Так как с измерением <code>(MonthYear, (NUMERIC, ASCENDING))</code> используется критерий сортировки, функция <code>Aggr</code> сравнивает количество клиентов за идущие подряд месяцы в виртуальной таблице с помощью сортировки месяцев в числовом порядке по возрастанию, а не в алфавитном порядке по возрастанию.</p>

Данные, используемые в примерах:

Создайте таблицу с элементами **Customer**, **Product**, **UnitPrice** и **UnitSales** в качестве измерений. Добавьте выражение в таблицу в качестве меры.

```
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|1|19
] (delimiter is '|');
```

См. также:

[р Базовые функции агрегирования \(страница 173\)](#)

### 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 диаграммы([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:** dual

#### Аргументы:

Аргумент	Описание
value Expression	С помощью функции можно найти значение выражения <b>value</b> , которое соответствует результату сортировки поля <b>sort_weight</b> .
sort-weight Expression	Выражение, содержащее данные для сортировки. Обнаружено первое (нижнее) значение элемента <b>sort_weight</b> , на основе которого определяется соответствующее значение выражения <b>value</b> . Если указать знак минуса перед элементом <b>sort_weight</b> , функция вернет последнее (самое высокое) отсортированное значение.
rank Expression	При указании для элемента <b>rank</b> значения «n» выше 1 будет получено n-ое отсортированное значение.
distinct	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

## 5 Функции в скриптах и выражениях диаграммы

### 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
<pre>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;</pre>	<p>Customer MyProductWithSmallestOrderByCustomer Astrida CC Betacab AA Canutility AA Divadip DD</p> <p>Функция выполняет сортировку значений UnitSales от наименьших к наибольшим, регистрируя значение параметра Customer с наименьшим значением параметра UnitSales, как наименьший заказ.</p> <p>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 CC соответствует наименьшему заказу (8) для клиента Canutility, а элемент DD соответствует наименьшему заказу (10) для клиента Divadip..</p>



Пример	Результат
<p>При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:</p> <pre>LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductwithLargestOrderByCustomer Resident Temp Group By Customer;</pre>	<pre>Customer MyProductwithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip -</pre> <p>Аргументу <code>sort_weight</code> предшествует знак минуса, поэтому с помощью функции элементы будут отсортированы от наибольших к наименьшим.</p> <p>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.</p>
<p>При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:</p> <pre>LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductwithSmallestOrderByCustomer Resident Temp Group By Customer;</pre>	<pre>Customer MyProductwithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip AA</pre> <p>Все действия будут выполняться так же, как и в предыдущем примере, но будет использоваться префикс <code>distinct</code>. При этом результат дубликата для Divadip будет проигнорирован, что позволит вернуть ненулевое значение.</p>

### 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:** dual

**Аргументы:**

Аргумент	Описание
value	Поле вывода. С помощью функции можно найти значение выражения <b>value</b> , которое соответствует результату сортировки поля <b>sort_weight</b> .
sort_weight	Поле ввода. Выражение, содержащее данные для сортировки. Обнаружено первое (нижнее) значение элемента <b>sort_weight</b> , на основе которого определяется соответствующее значение выражения <b>value</b> . Если указать знак минуса перед элементом <b>sort_weight</b> , функция вернет последнее (самое высокое) отсортированное значение.
rank	При указании для элемента <b>rank</b> 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Примеры и результаты:**

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20

## 5 Функции в скриптах и выражениях диаграммы

Customer	Product	UnitSales	UnitPrice
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

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 значением rank (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] )
```

## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
rank Expression	По умолчанию значение <b>rank</b> — 1, что соответствует наибольшему значению. При указании для <b>rank</b> значения 2 будет возвращено второе наибольшее значение. Если <b>rank</b> 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	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 ' ');  Max: LOAD Customer, Max(UnitSales) as MyMax, Resident Temp Group By Customer;	Customer	MyMax
	Astrida	18
	Betacab	5
	Canutility	8
При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:  LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;	Customer	MyMaxRank2
	Astrida	10
	Betacab	4
	Canutility	-

### Max — функция диаграммы

**Max()** находит наибольшее значение агрегированных данных. Если указать **rank** n, можно найти

## 5 Функции в скриптах и выражениях диаграммы

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:** число

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
rank	По умолчанию значение <b>rank</b> — 1, что соответствует наибольшему значению. При указании для <b>rank</b> значения 2 будет возвращено второе наибольшее значение. Если <b>rank</b> имеет значение 3, будет возвращено третье наибольшее значение, и т. д.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {,fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10

## 5 Функции в скриптах и выражениях диаграммы

Customer	Product	UnitSales	UnitPrice
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры	Результаты
<code>Max(UnitSales)</code>	Значение 10, поскольку это наибольшее значение в элементе <code>unitSales</code> .
Значение порядка вычисляется из числа проданных единиц в элементе <code>(unitSales)</code> , умноженного на стоимость единицы.  <code>Max (UnitSales*UnitPrice)</code>	Значение 150, поскольку это наибольшее значение, полученное в результате вычисления всех возможных значений элементов <code>(unitSales)*(UnitPrice)</code> .
<code>Max(UnitSales, 2)</code>	Значение 9, которое является вторым наибольшим значением.
<code>Max(TOTAL UnitSales)</code>	Значение 10, поскольку префикс <code>TOTAL</code> означает, что обнаружено наибольшее возможное значение без учета измерений диаграммы. Для диаграммы с элементом <code>Customer</code> в качестве измерения префикс <code>TOTAL</code> обеспечит возврат максимального значения по всему набору данных вместо максимального значения <code>UnitSales</code> для каждого клиента.
Выполнить выборку <code>Customer B</code> .  <code>Max({1} TOTAL UnitSales)</code>	Значение 10, независимо от сделанной выборки, поскольку выражение <code>Set Analysis {1}</code> определяет порядок записей для оценки в качестве элемента <code>ALL</code> , 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 (страница 177)

p *RangeMax* (страница 633)

### Min

Функция **Min()** возвращает наименьшее числовое значение агрегированных данных в выражении, как определено предложением **group by**. Если указать **rank** n, можно найти наименьшее n-ое значение.

**Синтаксис:**

```
Min ( expr [, rank] )
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
rank Expression	Значение <b>rank</b> по умолчанию равно 1, что соответствует наименьшему значению. При указании для <b>rank</b> значения 2 будет возвращено второе наименьшее значение. Если <b>rank</b> 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	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 ' ');  Min: LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;	Customer	MyMin
	Astrida	2
	Betacab	4
	Canutility	8
При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:  LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;	Customer	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:** число

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
rank	По умолчанию значение <b>rank</b> — 1, что соответствует наибольшему значению. При указании для <b>rank</b> значения 2 будет возвращено второе наибольшее значение. Если <b>rank</b> имеет значение 3, будет возвращено третье наибольшее значение, и т. д.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

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.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
<code>min(unitsSales)</code>	Значение 2, поскольку это наименьшее значение, не являющееся NULL, в элементе <code>unitsSales</code> .
Значение порядка вычисляется из числа проданных единиц в элементе <code>(unitsSales)</code> , умноженного на стоимость единицы.  <code>Min (UnitSales*UnitPrice)</code>	Значение 40, поскольку это наименьшее значение, не являющееся NULL, полученное в результате вычисления всех возможных значений элементов <code>(unitsSales)*(UnitPrice)</code> .
<code>min(unitsSales, 2)</code>	Значение 4, которое является вторым наименьшим значением (после значений NULL).
<code>min(TOTAL unitsSales)</code>	Значение 2, поскольку префикс TOTAL означает, что обнаружено наименьшее возможное значение без учета измерений диаграммы. Для диаграммы с элементом Customer в качестве измерения префикс TOTAL обеспечит возврат минимального значения по всему набору данных вместо минимального значения UnitSales для каждого клиента.
Выполнить выборку Customer B.  <code>min({1} TOTAL unitsSales)</code>	Значение 40, 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|unitsSales|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 (страница 177)  
*p RangeMin* (страница 637)

### 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:** dual

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.

Пример	Результат
<pre>Temp: LOAD * inline [ Customer Product OrderNumber UnitsSales 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;</pre>	<p>MyMostOftenSoldProduct</p> <p>AA</p> <p>поскольку AA — это единственный продукт, проданный несколько раз.</p>

### 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:** dual

#### 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	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {,fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

#### Примеры и результаты:

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

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
<code>Mode(UnitPrice)</code> Выполнить выборку <code>Customer A.</code>	Значение 15, поскольку это наиболее часто встречающееся значение в элементе <code>unitSales</code> .  Возвращает NULL (-). Одно значение встречается не чаще, чем другое.
<code>Mode(Product)</code> Выполните выборку <code>Customer A.</code>	Значение AA, поскольку это наиболее часто встречающееся значение в элементе <code>Product</code> .  Возвращает NULL (-). Одно значение встречается не чаще, чем другое.
<code>Mode (TOTAL UnitPrice)</code>	Значение 15, поскольку префикс TOTAL означает, что наиболее часто встречающимся значением все еще является 15, без учета измерений диаграммы.
Выполните выборку <code>Customer B.</code>  <code>Mode)({1}</code> <code>TOTAL UnitPrice)</code>	Значение 15, независимо от сделанной выборки, поскольку выражение <code>Set Analysis {1}</code> 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 (страница 233)*

*p Median — функция диаграммы (страница 268)*

### 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:** dual

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.

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 ' ');  Only: LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;	Customer  Astrida	MyUniqIDCheck  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:** dual

**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	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>



Используйте функцию *Only()*, если необходимо получить значение *NULL* в случае нескольких возможных значений в данных образца.

**Примеры и результаты:**

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

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
<code>only ({&lt;UnitPrice= {9}&gt;} Product)</code>	Значение BB, поскольку это единственный элемент Product, у которого элемент unitPrice равен 9.
<code>only({&lt;Product= {DD}&gt;} Customer)</code>	Значение B, поскольку единственный элемент Customer, продающий Product, называется «DD».
<code>only ({&lt;UnitPrice= {20}&gt;} unitSales)</code>	Число элементов unitSales, где элемент unitPrice, равный 20, составляет 2, поскольку есть только одно значение элемента unitSales, где unitPrice = 20.
<code>only ({&lt;UnitPrice= {15}&gt;} unitSales)</code>	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|1|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:** число

**Аргументы:**

Аргумент	Описание
distinct	Если слово <b>distinct</b> 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.

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 ' ');  Sum: LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;	Customer  Astrida  Betacab  Canutility	MySum  39  9  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:** число

#### 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.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
DISTINCT	<p>Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.</p> <div><i>Несмотря на то, что префикс <b>DISTINCT</b> 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.</i></div>
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

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	Результаты
Sum(UnitSales)	38. Итого значений в элементе unitSales.
Sum(UnitSales*UnitPrice)	505. Итого элемента unitPrice, умноженное на агрегированный элемент unitSales.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
Sum (TOTAL unitsSales*UnitPrice)	Значение 505 для всех строк в таблице, а также итоговое значение, поскольку префикс TOTAL означает, что сумма по-прежнему равна 505, без учета измерений диаграммы.
Выполните выборку Customer B.  Sum({1} TOTAL unitsSales*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|unitsSales|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

## 5 Функции в скриптах и выражениях диаграммы

---

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 число

#### Аргументы:

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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	Результат
<pre>Temp: LOAD * inline [ Customer Product OrderNumber UnitsSales 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;</pre>	<pre>Customer OrdersByCustomer Astrida 3 Betacab 3 Canutility 2 Divadip 2</pre> <p>При условии, что измерение Customer включено в таблицу на листе, в противном случае результатом для OrdersByCustomer будет 3, 2.</p>
<p>При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:</p>	<pre>TotalOrderNumber 10</pre>
<p>При условии, что таблица <b>Temp</b> загружается, как в первом примере:</p> <pre>LOAD Count(distinct OrderNumber) as TotalOrdersNumber Resident Temp;</pre>	<pre>TotalOrderNumber 9</pre> <p>Поскольку существуют два значения элемента OrderNumber с одинаковым значением (1).</p>

### 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 число

#### 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.

## 5 Функции в скриптах и выражениях диаграммы


Аргумент	Описание
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

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.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
Count(OrderNumber)	10, поскольку существует 10 полей, которые могут иметь значение для элемента OrderNumber, а также учитываются все записи, даже пустые. <div> «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.</div>
Count (Customer)	Значение 10, поскольку элемент Count оценивает число вхождений во всех полях.
Count (DISTINCT [Customer])	Значение 4, поскольку при использовании префикса 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|UnitsSales|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 число

### Аргументы:

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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	Результат
<pre>Temp: LOAD * inline [ Customer Product OrderNumber UnitsSales 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;</pre>	<pre>Customer MissingOrdersByCustomer Astrida 0 Betacab 1 Canutility 2 Divadip 0  Второй оператор дает следующее:  TotalMissingCount 3 в таблице с этим измерением.</pre>

Пример	Результат
<p>При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:</p> <pre>LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;</pre>	<pre>TotalMissingCountDistinct 1</pre> <p>Поскольку одним отсутствующим значением является только один элемент OrderNumber.</p>

### 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 число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {,fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

#### Примеры и результаты:

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15

## 5 Функции в скриптах и выражениях диаграммы

Customer	Product	OrderNumber	UnitSales	Unit Price
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

Пример	Результат
MissingCount([OrderNumber])	<p>Значение 3, поскольку 3 из 10 полей OrderNumber являются пустыми</p> <div>  <p>«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.</p> </div>
MissingCount([OrderNumber])/MissingCount({1} Total [OrderNumber])	<p>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.</p>

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 число

#### Аргументы:

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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	Результат
<pre>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;</pre>	<p>Customer NullOrdersByCustomer Astrida 0 Betacab 0 Canutility 1</p> <p>Второй оператор дает следующее:</p> <p>TotalNullCount 1</p> <p>в таблице с этим измерением, поскольку единственная запись содержит нулевое значение.</p>

### 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 число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

Пример	Результат
NullCount ([OrderNumber])	Значение 1, поскольку введено нулевое значение с помощью элемента NullInterpret во встроенном операторе <b>LOAD</b> .

Данные, используемые в примере:

```
Set NULLINTERPRET = NULL;
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitsSales|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 число

**Аргументы:**

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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	Результат
Temp: LOAD * inline [ Customer Product OrderNumber UnitsSales 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;	Customer      NumericCountByCustomer Astrida        3 Betacab        2 Canutility     0 Divadip        2
LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;	Второй оператор обеспечивает: TotalNumericCount 7 в таблице с этим измерением.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:  <code>LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;</code>	TotalNumericCountDistinct 6 Поскольку существует один элемент OrderNumber, который дублирует другой элемент, поэтому результатом будет значение 6. Это элементы не являются дубликатами.

### 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 число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово <b>TOTAL</b> 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 <b>TOTAL [&lt;fld {,fld}&gt;]</b> , где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

#### Примеры и результаты:


Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16



## 5 Функции в скриптах и выражениях диаграммы

Customer	Product	OrderNumber	UnitSales	Unit Price
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.

Пример	Результат
NumericCount ([OrderNumber])	<p>Значение 7, поскольку 3 из 10 полей в элементе OrderNumber пустые.</p> <div>  <p>«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.</p> </div>
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 число

#### Аргументы:

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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	Результат										
<pre>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;</pre>	<table> <tr> <th>Customer</th><th>ProductTextCount</th></tr> <tr> <td>Astrida</td><td>3</td></tr> <tr> <td>Betacab</td><td>3</td></tr> <tr> <td>Canutility</td><td>2</td></tr> <tr> <td>Divadip</td><td>2</td></tr> </table>	Customer	ProductTextCount	Astrida	3	Betacab	3	Canutility	2	Divadip	2
Customer	ProductTextCount										
Astrida	3										
Betacab	3										
Canutility	2										
Divadip	2										
<pre>LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;</pre>	<table> <tr> <th>Customer</th><th>OrderNumberTextCount</th></tr> <tr> <td>Astrida</td><td>0</td></tr> <tr> <td>Betacab</td><td>1</td></tr> <tr> <td>Canutility</td><td>2</td></tr> <tr> <td>Divadip</td><td>0</td></tr> </table>	Customer	OrderNumberTextCount	Astrida	0	Betacab	1	Canutility	2	Divadip	0
Customer	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 число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

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

Пример	Результат
TextCount ([Product])	<p>Значение 10, поскольку все из 10 полей в элементе Product текстовые.</p> <div><p>«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.</p></div>

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
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|UnitsSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|1|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

### Функции финансового агрегирования

В этом разделе описаны функции агрегирования для финансовых операций в отношении платежей и денежного потока.

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

### Функции финансового агрегирования в скрипте загрузки данных

#### IRR

Функция **IRR()** возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами выражений, повторяемых в нескольких записях так, как это определено предложением group by.

**IRR** (expression)

#### XIRR

Функция **XIRR()** возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Все платежи учитываются на основе года с 365 днями.

**XIRR** (valueexpression, dateexpression )

### NPV

**NPV()** возвращает агрегированную чистую текущую стоимость вложения на основе **discount\_rate** за период и ряда будущих платежей (отрицательные значения) и поступлений (положительные значения), представленных числами в элементе **value**, повторяемом в нескольких записях так, как это определено предложением group by. Предполагается, что платежи и поступления происходят в конце каждого периода.

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

### XNPV

Функция **XNPV()** возвращает агрегированную чистую текущую стоимость для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Rate — это процентная ставка за период. Все платежи учитываются на основе года с 365 днями.

```
XNPV (rate, valueexpression, dateexpression)
```

## Функции финансового агрегирования в выражениях диаграмм

Эти функции финансового агрегирования можно использовать в диаграммах.

### IRR

**IRR()** возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами в выражении, выданном элементом **value**, повторяемом в измерениях диаграммы.

```
IRR — функция диаграммы([TOTAL [<fld {,fld}>]] value)
```

### NPV

Функция **NPV()** возвращает агрегированную чистую стоимость инвестиций на основе скидки **discount\_rate** за период, серии будущих платежей (отрицательные значения) и дохода (положительные значения), представленных числами в элементе **value**, повторяемом в измерениях диаграммы. Предполагается, что платежи и поступления происходят в конце каждого периода.

```
NPV — функция диаграммы([TOTAL [<fld {,fld}>]] discount_rate, value)
```

### XIRR

**XIRR()** возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

```
XIRR — функция диаграммы (страница 221) ([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:** число

#### 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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

### Примеры и результаты:

Пример	Результат	
<b>Cashflow:</b> 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 ' ');  <b>Cashflow1:</b> LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;	Year  2013	IRR2013  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:** число

#### Аргументы:

Аргумент	Описание
value	Выражение или поле, содержащее данные для измерения.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {,fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>




### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

### Примеры и результаты:

Пример	Результат
IRR (Payments)	0,1634  Предполагается, что платежи являются периодическими, например ежемесячными.  <div> Поле Date используется в примере XIRR, где платежи могут быть не периодическими, если указываются даты, в которые совершаются платежи.</div>

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 (страница 221)

p *Aggr* — функция диаграммы (страница 169)

## NPV

**NPV()** возвращает агрегированную чистую текущую стоимость вложения на основе **discount\_rate** за период и ряда будущих платежей (отрицательные значения) и поступлений (положительные значения), представленных числами в элементе **value**, повторяемом в нескольких записях так, как это определено предложением group by. Предполагается, что платежи и поступления происходят в конце каждого периода.

### Синтаксис:

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

## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** числовой. Результат имеет числовой денежный формат по умолчанию.

### Аргументы:

Аргумент	Описание
discount_rate	<b>discount_rate</b> — это льготный тариф за какой-либо период.
value	Выражение или поле, содержащее данные для измерения.

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

### Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат		
Cashflow: LOAD 2013 as Year, * inline [ Date Discount Payments 2013-01-01 0.1 -10000 2013-03-01 0.1 3000 2013-10-30 0.1 4200 2014-02-01 0.2 6800 ] (delimiter is ' ');  Cashflow1: LOAD Year,NPV(0.2, Payments) as NPV1_2013 Resident Cashflow Group By Year;	Year		NPV1_2013
	2013		-\$540.12
При условии, что таблица <b>Cashflow</b> загружается, как в предыдущем примере: LOAD Year,NPV(Discount, Payments) as NPV2_2013 Resident Cashflow Group By Year, Discount; Обратите внимание, что предложение Group By сортирует результаты по элементам Year и Discount. Первый аргумент discount_rate дан в виде поля (Discount), а не специального номера, и, таким образом, требуется второй критерий сортировки. Поле может содержать различные значения, поэтому агрегированные записи должны быть отсортированы, чтобы обеспечить различные значения Year и Discount.	Year	Discount	NPV2_2013
	2013	0,1	-3456,05 \$
	2013	0,2	5666,67 \$

### NPV — функция диаграммы

Функция **NPV()** возвращает агрегированную чистую стоимость инвестиций на основе скидки **discount\_rate** за период, серии будущих платежей (отрицательные значения) и дохода (положительные значения), представленных числами в элементе **value**, повторяемом в измерениях диаграммы. Предполагается, что платежи и поступления происходят в конце каждого периода.

#### Синтаксис:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

**Возвращаемые типы данных:** число Результат имеет числовой денежный формат по умолчанию.

#### Аргументы:

Аргумент	Описание
discount_rate	<b>discount_rate</b> — это льготный тариф за какой-либо период.
value	Выражение или поле, содержащее данные для измерения.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {,fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p> <p>После префикса <b>TOTAL</b> 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.</p>

#### 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 функцию расширенного агрегирования **Aggr**

## 5 Функции в скриптах и выражениях диаграммы

вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

### 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 (страница 224)

p *Aggr* — функция диаграммы (страница 169)

## 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:** число

### Аргументы:

Аргумент	Описание
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе <b>date</b> .
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе <b>pmt</b> .

### 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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат	
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 ' ');	Year	XIRR2013
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:** число

### Аргументы:

Аргумент	Описание
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе <b>date</b> .
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе <b>pmt</b> .

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### 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 функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### 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 (страница 216)

p *Aggr* — функция диаграммы (страница 169)

## XNPV

Функция **XNPV()** возвращает агрегированную чистую текущую стоимость для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Rate —

## 5 Функции в скриптах и выражениях диаграммы

это процентная ставка за период. Все платежи учитываются на основе года с 365 днями.

### Синтаксис:

```
XNPV(discount_rate, pmt, date)
```

**Возвращаемые типы данных:** числовой. Результат имеет числовой денежный формат по умолчанию. .

### Аргументы:

Аргумент	Описание
discount_rate	<b>discount_rate</b> — это льготный тариф за какой-либо период.
pmt	Выражение или поле, содержащее данные для измерения.
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе <b>pmt</b> .

### 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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат	
Cashflow: LOAD 2013 as Year, * inline [ Date Discount Payments 2013-01-01 0.1 -10000 2013-03-01 0.1 3000 2013-10-30 0.1 4200 2014-02-01 0.2 6800 ] (delimiter is ' ');  Cashflow1: LOAD Year,XNPV(0.2, Payments, Date) as XNPV1_2013 Resident Cashflow Group By Year;	Year  2013	XNPV1_2013  \$2104.37

Пример	Результат		
При условии, что таблица <b>Cashflow</b> загружается, как в предыдущем примере: LOAD Year,XNPV(Discount, Payments, Date) as XNPV2_2013 Resident Cashflow Group By Year, Discount; Обратите внимание, что предложение Group By сортирует результаты по элементам Year и Discount. Первый аргумент discount_rate дан в виде поля (Discount), а не специального номера, и, таким образом, требуется второй критерий сортировки. Поле может содержать различные значения, поэтому агрегированные записи должны быть отсортированы, чтобы обеспечить различные значения Year и Discount.	Year	Discount	XNPV2_2013
	2013	0,1	-3164,35 \$
	2013	0,2	6800,00 \$

### XNPV — функция диаграммы

**XNPV()** возвращает агрегированную чистую стоимость для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

#### Синтаксис:

```
XNPV ([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

**Возвращаемые типы данных:** число Результат имеет числовой денежный формат по умолчанию.

#### Аргументы:

Аргумент	Описание
discount_rate	<b>discount_rate</b> — это льготный тариф за какой-либо период.
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе <b>date</b> .
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе <b>pmt</b> .



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### 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 функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### 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:

р *NPV* — функция диаграммы (страница 219)

р *Aggr* — функция диаграммы (страница 169)

## 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 значение, соответствующее квантилю агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

```
Fractile (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

## 5 Функции в скриптах и выражениях диаграммы

---

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**

Функция **LINEST\_SSREG()** возвращает агрегированную остаточную сумму квадратов линейной регрессии, определенной уравнением  $y=mx+b$  для серии координат, представленных парными

## 5 Функции в скриптах и выражениях диаграммы

---

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 агрегированное значение median значений 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.

## 5 Функции в скриптах и выражениях диаграммы

---

### 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, выданном выражением, повторяемым в измерениях диаграммы.

```
Fractile — 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 ( $r^2/(1-r^2)$ ) линейной регрессии,

---

## 5 Функции в скриптах и выражениях диаграммы

---

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

## 5 Функции в скриптах и выражениях диаграммы

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(**{**[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**)

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:** число

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
DISTINCT	Если слово <b>distinct</b> 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.

Пример	Результат
<pre>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;</pre>	<pre>Customer MyAverageSalesByCustomer Astrida 48.916667 Betacab 44.916667 Canutility 56.916667 Divadip 63.083333</pre> <p>Это можно проверить на листе путем создания таблицы, включая меру: <code>sum(Sales)/12</code></p>
<p>При условии, что таблица <b>Temp</b> загружается, как в предыдущем примере:</p> <pre>LOAD Customer, Avg(DISTINCT Sales) as MyAvgSalesDistinct Resident Temp Group By Customer;</pre>	<pre>Customer MyAverageSalesByCustomer Astrida 43.1 Betacab 43.909091 Canutility 55.909091 Divadip 61</pre> <p>Учитываются только уникальные значения. Поделите итоговое значение на количество неповторяющихся значений.</p>



### 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:** число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

#### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

#### Примеры и результаты:

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

## 5 Функции в скриптах и выражениях диаграммы

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
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

Customer	Sum([Sales])	Avg([Sales])	Avg(TOTAL Sales)	Avg(DISTINCT Sales)	Avg({1} TOTAL Sales)
	2566	53.46	53,458333	51,862069	53,458333
Astrida	587	48.92	53,458333	43,1	53,458333
Betacab	539	44.92	53,458333	43,909091	53,458333
Canutility	683	56.92	53,458333	55,909091	53,458333
Divadip	757	63.08	53,458333	61	53,458333

Пример	Результат
Avg (Sales)	Для таблицы, включающей измерение customer и меру Avg([Sales]), если показано значение <b>Итоги</b> , результат будет 2566.
Avg ([TOTAL Sales])	53,458333 для всех значений элемента customer, поскольку префикс TOTAL означает, что измерения игнорируются.
Avg (DISTINCT Sales)	51,862069 для итогового значения, поскольку использование префикса Distinct означает, что оцениваются только уникальные значения в поле Sales для каждого элемента customer.

Данные, используемые в примерах:

```
Monthnames:
LOAD * INLINE [
Month, 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 (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 '|');
```

## 5 Функции в скриптах и выражениях диаграммы

---

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

---

**См. также:**

*p Aggr* — функция диаграммы (страница 169)

### 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:** число

**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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат
<pre>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;</pre>	<p>В таблице с измерением correl_salary, результат вычисления Correl() в скрипте загрузки данных будет показан как: 0,9270611</p>

### 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:** число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### Примеры и результаты:

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.
Correl (TOTAL Age, Salary))	<p>0,927. Этот и следующие результаты показаны в формате с тремя знаками после десятичной запятой для удобства считывания.</p> <p>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.</p>
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 (страница 169)

*p Avg* — функция диаграммы (страница 233)

*p RangeCorrel* (страница 625)

### Fractile

Функция **Fractile()** находит значение, соответствующее квантилю агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

**Синтаксис:**

```
Fractile(expr, fraction)
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат
<pre>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;</pre>	<p>В таблице с измерениями type и myFractile результаты вычислений Fractile() в скрипте загрузки данных будут показаны как:</p> <p>type myFractile Comparison 27.5 Observation 36</p>

### Fractile — функция диаграммы

Функция **Fractile()** находит значение, соответствующее квантилю агрегированных данных в диапазоне, выданном выражением, повторяемым в измерениях диаграммы.

#### Синтаксис:

```
Fractile ([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

**Возвращаемые типы данных:** число

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
fraction	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.

## 5 Функции в скриптах и выражениях диаграммы

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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

### Примеры и результаты:

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

Пример	Результат
Fractile (Sales, 0.75)	Для таблицы, включающей измерение customer и меру Fractile([Sales]), если показано значение <b>Итоги</b> , 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 префикс TOTAL означает, что измерения игнорируются.



Пример	Результат
Fractile (DISTINCT Sales, 0.75)	70 для итогового значения, поскольку использование префикса DISTINCT означает, что оцениваются только уникальные значения в поле Sales для каждого элемента Customer.

Данные, используемые в примерах:

```
Monthnames:
LOAD * INLINE [
Month, 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 (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 '|');
```

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

**См. также:**

p Aggr — функция диаграммы (страница 169)

### Kurtosis

Функция **Kurtosis()** возвращает эксцесс данных в выражении в нескольких записях, как это определено предложением **group by**.

**Синтаксис:**

```
Kurtosis([distinct ] expr )
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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.

Пример	Результат
<pre>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;</pre>	<p>В таблице с измерениями Type, MyKurtosis1 и MyKurtosis2 результаты вычислений Kurtosis() в скрипте загрузки данных будут показаны как:</p> <pre>Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144</pre>

### 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:** число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

#### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

#### Примеры и результаты:

Type	Value
Comparison	<div>2 2 3 3 1 1 1 3 3 1 2 3 2 1 2 1 3 2 3 2</div> <div>7 8 1 9 4 9 7</div>

## 5 Функции в скриптах и выражениях диаграммы

Type	Value																			
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Пример	Результат
kurtosis (value)	Если для таблицы, включающей измерение type и меру kurtosis(value), показано значение <b>Итоги</b> , 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 префикс 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 (страница 233)*

### 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:** число

### 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)	<p>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.</p> <p>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.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

## 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:** число

**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_const, x0_const	<p>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.</p> <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

р *Avg — функция диаграммы (страница 233)*

### 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: число

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)	<p>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.</p> <p>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.</p>

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

### 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.


## 5 Функции в скриптах и выражениях диаграммы

### Синтаксис:

```
LINEST_DF ([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value [, y0_const [, x0_const]])
```

**Возвращаемые типы данных:** число

### 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	<p>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.</p> <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.



## 5 Функции в скриптах и выражениях диаграммы

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

р *Avg — функция диаграммы (страница 233)*

## 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  $x\text{-expression}$  и  $y\text{-expression}$ , повторяемых в нескольких записях так, как это определено выражением **group by**.

### Синтаксис:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

**Возвращаемые типы данных:** число

### 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)	<p>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.</p> <p>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.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

### LINEST\_F — функция диаграммы


Функция **LINEST\_F()** возвращает агрегированное статистическое  $F(r^2/(1-r^2))$  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:** число

#### 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	<p>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.</p> <div>  <p>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.</p> </div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

р *Avg — функция диаграммы (страница 233)*

## 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:** число

### Аргументы:

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений <i>y</i> для измерения.
x_value	Выражение или поле, содержащее диапазон значений <i>x</i> для измерения.
y(0), x(0)	<p>Дополнительное значение <i>y0</i> можно указать путем принудительного прохождения линии регрессии через ось <i>y</i> в определенной точке. Указав <i>y0</i> и <i>x0</i>, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату.</p> <p>Если значения <i>y0</i> и <i>x0</i> не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если <i>y0</i> и <i>x0</i> указаны, используется одна пара значений.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

### 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:** число

**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. <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

р *Avg — функция диаграммы (страница 233)*

## 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:** число

### 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:

р *Примеры использования функций `linest` (страница 281)*

## 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:** число

### 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	<p>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.</p> <div>  <p>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.</p> </div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*  
 р *Avg — функция диаграммы (страница 233)*

## 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:** число

### 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)	<p>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.</p> <p>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.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

## 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:** число

**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	<p>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.</p> <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

р Avg — функция диаграммы (страница 233)

### 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: число

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)	<p>Дополнительное значение <math>y_0</math> можно указать путем принудительного прохождения линии регрессии через ось <math>y</math> в определенной точке. Указав <math>y_0</math> и <math>x_0</math>, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату.</p> <p>Если значения <math>y_0</math> и <math>x_0</math> не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если <math>y_0</math> и <math>x_0</math> указаны, используется одна пара значений.</p>

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

### 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.


## 5 Функции в скриптах и выражениях диаграммы

### Синтаксис:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

**Возвращаемые типы данных:** число

### 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	<p>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.</p> <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

## 5 Функции в скриптах и выражениях диаграммы

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

**См. также:**

р *Примеры использования функций `linest` (страница 281)*

р *Avg — функция диаграммы (страница 233)*

### LINEST\_SEY

Функция **LINEST\_SEY()** возвращает агрегированную стандартную ошибку оценки у 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:** число

**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)	<p>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.</p> <p>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.</p>

**Ограничения:**

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

**См. также:**

р *Примеры использования функций `linest` (страница 281)*

### LINEST\_SEY — функция диаграммы


Функция **LINEST\_SEY()** возвращает агрегированную стандартную ошибку значения у 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:** число

#### 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	<p>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.</p> <div>  <p>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.</p> </div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций `linest` (страница 281)*

р *Avg — функция диаграммы (страница 233)*

## LINEST\_SSREG

Функция **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:** число

### 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)	<p>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.</p> <p>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.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

р *Примеры использования функций linest (страница 281)*

### 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:** число

**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. <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций linest (страница 281)*

р *Avg — функция диаграммы (страница 233)*

## 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:** число

### 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)	<p>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.</p> <p>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.</p>

### Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций `linest` (страница 281)*

## 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:** число

### 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.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
y0, x0	<p>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.</p> <div> 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.</div>
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

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:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

### См. также:

р *Примеры использования функций **linest** (страница 281)*

р *Avg — функция диаграммы (страница 233)*

### Median

Функция **Median()** возвращает агрегированное значение median значений в выражении в нескольких записях, как это определено предложением **group by**.

#### Синтаксис:

```
Median (expr)
```

**Возвращаемые типы данных:** число

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

#### Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу, используя `type` и `myMedian` в качестве измерений.

Пример	Результат
<pre>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 ' ');  Median1: LOAD Type, Median(Value) as myMedian Resident Table1 Group By Type;</pre>	<p>Результаты вычисления Median() выглядят следующим образом:</p> <ul style="list-style-type: none"><li>• <code>type</code> — <code>myMedian</code></li><li>• <code>comparison</code> — 2.5</li><li>• <code>observation</code> — 26.5</li></ul>

### 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:** число

#### 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {, fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

#### Ограничения:

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

#### Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу, установив измерение `type` и меру `median(value)`.

Необходимо активировать элемент `totals` в свойствах таблицы.

Пример	Результат
<pre>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 ' ');</pre>	<p>Значения медианы для следующих элементов составляет:</p> <ul style="list-style-type: none"> <li>Totals — 19</li> <li>Comparison — 2.5</li> <li>Observation — 26.5</li> </ul>

### См. также:

*p Avg* — функция диаграммы (страница 233)

## Skew

Функция **Skew()** возвращает асимметрию выражения в нескольких записях, как это определено предложением **group by**.

### Синтаксис:

```
Skew([ distinct] expr)
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
DISTINCT	Если слово <b>distinct</b> 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.

Пример	Результат
<pre>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;</pre>	<p>Результаты вычисления <code>Skew()</code> выглядят следующим образом:</p> <ul style="list-style-type: none"><li>• <code>Type</code> — <code>MySkew</code></li><li>• <code>Comparison</code> — 0.86414768</li><li>• <code>observation</code> — 0.32625351</li></ul>

### 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:** число

**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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

**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
<pre>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 ' ');</pre>	<p>Результаты вычисления Skew(Value) выглядят следующим образом:</p> <ul style="list-style-type: none"> <li>• Total — 0.23522195</li> <li>• Comparison — 0.86414768</li> <li>• observation — 0.32625351</li> </ul>

### См. также:

*p Avg* — функция диаграммы (страница 233)

### 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:** число

### Аргументы:

Аргумент	Описание
<i>expr</i>	Выражение или поле, содержащее данные для измерения.
<i>distinct</i>	Если слово <b>distinct</b> 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.

Пример	Результат
<pre>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;</pre>	<p>Результаты вычисления <code>Stdev()</code> выглядят следующим образом:</p> <ul style="list-style-type: none"><li>• <code>Type</code> — <code>MyStdev</code></li><li>• <code>Comparison</code> — 14.61245</li><li>• <code>Observation</code> — 12.507997</li></ul>

### 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:** число

**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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

**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
<pre>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 ' ');</pre>	<p>Результаты вычисления Stdev(Value) выглядят следующим образом:</p> <ul style="list-style-type: none"> <li>• Total — 15.47529</li> <li>• Comparison — 14.61245</li> <li>• observation — 12.507997</li> </ul>

### См. также:

р Avg — функция диаграммы (страница 233)

р STEYX — функция диаграммы (страница 279)

## Sterr

Функция **Sterr()** возвращает агрегированную стандартную ошибку ( $\text{stdev}/\sqrt{n}$ ) для серии значений, представленных выражением, повторяемым в нескольких записях так, как это определено предложением **group by**.

### Синтаксис:

```
Sterr ([distinct] expr)
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово <b>distinct</b> 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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат
<pre>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;</pre>	<p>В таблице с измерениями type и mySterr результаты вычисления Sterr() в скрипте загрузки данных будут показаны как:</p> <pre>type mySterr Comparison 3.2674431 Observation 2.7968733</pre>

### Sterr — функция диаграммы

Функция **Sterr()** находит значение стандартной ошибки среднего значения ( $\text{stdev}/\sqrt{n}$ ) для серии значений, агрегированных в выражении, повторяемом в измерениях диаграммы.

### Синтаксис:

```
Sterr ([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

**Возвращаемые типы данных:** число

**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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

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
<pre>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 ' ');</pre>	<p>Результаты вычисления <code>Sterr(Value)</code> выглядят следующим образом:</p> <ul style="list-style-type: none"> <li>• Total — 2.4468583</li> <li>• Comparison — 3.2674431</li> <li>• observation — 2.7968733</li> </ul>

### См. также:

`p Avg` — функция диаграммы (страница 233)

`p STEYX` — функция диаграммы (страница 279)

## 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:** число

### Аргументы:

Аргумент	Описание
<code>y_value</code>	Выражение или поле, содержащее диапазон значений <code>y</code> для измерения.
<code>x_value</code>	Выражение или поле, содержащее диапазон значений <code>x</code> 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. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Пример	Результат
<pre>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;</pre>	В таблице с измерением <code>mysteyx</code> результат вычисления <code>STEYX()</code> 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:** число

**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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Выражение не должно содержать функции агрегирования, кроме внутреннего агрегирования, содержащего префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать функцию расширенного агрегирования **Aggr** вместе с вычисляемыми измерениями.

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
<pre>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 ' ');</pre>	Результат вычисления STEYX(KnownY,KnownX) равен 2,071 (если форматирование числа установлено на значение "3 десятичных знака".)

### См. также:

*p Avg* — функция диаграммы (страница 233)

*p Sterr* — функция диаграммы (страница 276)

### 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 индивидуальных темах функций диаграммы и скрипта linest.

### Загрузка данных образца

Выполните следующие действия.

1. Создайте новое приложение.
2. Введите в редакторе загрузки данных следующее:

```
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;
```

3. Щелкните элемент **I** , чтобы загрузить данные.

### Отображение результатов из вычислений скрипта загрузки данных

1. Выполните следующие действия.  
В редакторе загрузки данных щелкните элемент **”** , чтобы перейти в вид приложения, создайте новый лист и откройте его.
2. Щелкните команду **@ Изменить** , чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте в качестве столбцов следующее:
  - Linest\_B
  - Linest\_DF
  - Linest\_F
  - Linest\_M
  - Linest\_R2
  - Linest\_SEB
  - Linest\_SEM
  - Linest\_SEY
  - Linest\_SSREG
  - Linest\_SSRESID

Таблица, содержащая результаты вычислений linest, выполненных в скрипте загрузки данных, должна выглядеть так:

## 5 Функции в скриптах и выражениях диаграммы

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

### Создание визуализаций функции диаграммы linest

Выполните следующие действия.

1. В редакторе загрузки данных щелкните элемент ” , чтобы перейти в вид приложения, создайте новый лист и откройте его.
2. Щелкните команду **@ Изменить**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте линейный график, а из раздела **Поля** — элемент X как измерение и Sum(Y) как меру.  
Линейный график создан для представления графика элемента X, нанесенного напротив элемента Y, из которого вычисляются функции linest.
4. Из раздела **Диаграммы** добавьте таблицу со следующим в качестве измерения:  
valueList('Linest\_b', 'Linest\_df', 'Linest\_f', 'Linest\_m', 'Linest\_r2', 'Linest\_SEB', 'Linest\_SEM', 'Linest\_SEY', 'Linest\_SSREG', 'Linest\_SSRESID')  
В данном случае используется функция синтетических измерений для создания меток для измерений с именами функций linest. Для экономии места метку можно изменить на **Linest functions**.
5. 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) )  
В данном случае отображается 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

Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788

Linest functions	Linest function results
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

### Статистические функции тестирования

В этом разделе описаны функции для статистических тестов, разделенных на три категории. 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 Т-критериев

Функции t-критерия используются для статистического исследования двух генеральных средних. Т-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.

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-<sup>2</sup> для одной или двух серий значений.

```
Chi2Test_chi2(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.

```
Chi2Test_df(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:

р *Функции T-критериев (страница 288)*

р *Функции Z-критериев (страница 323)*

### Chi2Test\_chi2

Функция **Chi2Test\_chi2()** возвращает агрегированное значение критерия Хи-<sup>2</sup> 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<sup>2</sup>-критерия имеют одинаковые аргументы.*

### Синтаксис:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
actual_value	Наблюдаемое значение данных при указанных элементах <b>col</b> и <b>row</b> .
expected_value	Ожидаемое значение для распределения при указанных элементах <b>col</b> и <b>row</b> .

### 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:

р *Примеры использования функций chi2-test в диаграммах (страница 339)*

р *Примеры использования функций chi2-test в скрипте загрузки данных (страница 342)*

### 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<sup>2</sup>-критерия имеют одинаковые аргументы.*

### Синтаксис:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.
actual_value	Наблюдаемое значение данных при указанных элементах <b>col</b> и <b>row</b> .
expected_value	Ожидаемое значение для распределения при указанных элементах <b>col</b> и <b>row</b> .

### 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:

*р Примеры использования функций  $\chi^2$ -test в диаграммах (страница 339)*

*р Примеры использования функций  $\chi^2$ -test в скрипте загрузки данных (страница 342)*

### Chi2Test\_p — функция диаграммы

Функция **Chi2Test\_p()** возвращает агрегированное р-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:** число

### Аргументы:

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.
actual_value	Наблюдаемое значение данных при указанных элементах <b>col</b> и <b>row</b> .
expected_value	Ожидаемое значение для распределения при указанных элементах <b>col</b> и <b>row</b> .

### 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:

р *Примеры использования функций chi2-test в диаграммах (страница 339)*

р *Примеры использования функций chi2-test в скрипте загрузки данных (страница 342)*

## Функции Т-критериев

Функции t-критерия используются для статистического исследования двух генеральных средних. Т-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.

См.: *Создание типичного отчета t-test (страница 344)*

### Т-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.

```
TTest_conf ( grp, value [, sig[, eq_var]])
```

#### ttest\_df

Функция **TTest\_df()** возвращает агрегированное значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений.

```
TTest_df (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.

```
TTest_dif (grp, value)
```



ttest\_lower

Функция **TTest\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
TTest_lower (grp, value [, sig[, eq_var]])
```

ttest\_sig

Функция **TTest\_sig()** возвращает агрегированное значение двуххвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

```
TTest_sig (grp, value [, eq_var])
```

ttest\_sterr

Функция **TTest\_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений.

```
TTest_sterr (grp, value [, eq_var])
```

ttest\_t

Функция **TTest\_t()** возвращает агрегированное t-значение для двух независимых серий значений.

```
TTest_t (grp, value [, eq_var])
```

ttest\_upper

Функция **TTest\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
TTest_upper (grp, value [, sig [, eq_var]])
```

### Т-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.

```
TTestw_conf (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.

```
TTestw_df (weight, grp, value [, eq_var])
```

ttestw\_dif

Функция **TTestw\_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

## 5 Функции в скриптах и выражениях диаграммы

---

```
TTestw_dif ( weight, grp, value)
```

ttestw\_lower

Функция **TTestw\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
TTestw_lower (weight, grp, value [, sig[, eq_var]])
```

ttestw\_sig

Функция **TTestw\_sig()** возвращает агрегированное значение двуххвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

```
TTestw_sig ( 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.

```
TTestw_sterr (weight, grp, value [, eq_var])
```

ttestw\_t

Функция **TTestw\_t()** возвращает агрегированное t-значение для двух независимых серий значений.

```
TTestw_t (weight, grp, value [, eq_var])
```

ttestw\_upper

Функция **TTestw\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

### Т-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.

```
TTest1_conf (value [, sig])
```

ttest1\_df

Функция **TTest1\_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

```
TTest1_df (value)
```

ttest1\_dif

Функция **TTest1\_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений.

## 5 Функции в скриптах и выражениях диаграммы

---

**TTest1\_dif** (value)

ttest1\_lower

Функция **TTest1\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

**TTest1\_lower** (value [, sig])

ttest1\_sig

Функция **TTest1\_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

**TTest1\_sig** (value)

ttest1\_sterr

Функция **TTest1\_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

**TTest1\_sterr** (value)

ttest1\_t

Функция **TTest1\_t()** возвращает агрегированное t-значение для серии значений.

**TTest1\_t** (value)

ttest1\_upper

Функция **TTest1\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

**TTest1\_upper** (value [, sig])

### Т-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 функция **numeric**, которая возвращает агрегированное значение доверительного интервала для серии значений.

**TTest1w\_conf** (weight, value [, sig])

ttest1w\_df

Функция **TTest1w\_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

**TTest1w\_df** (weight, value)

ttest1w\_dif

Функция **TTest1w\_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента

---

## 5 Функции в скриптах и выражениях диаграммы

---

для серии значений.

```
TTest1w_dif (weight, value)
```

ttest1w\_lower

Функция **TTest1w\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

```
TTest1w_lower (weight, value [, sig])
```

ttest1w\_sig

Функция **TTest1w\_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

```
TTest1w_sig (weight, value)
```

ttest1w\_sterr

Функция **TTest1w\_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

```
TTest1w_sterr (weight, value)
```

ttest1w\_t

Функция **TTest1w\_t()** возвращает агрегированное t-значение для серии значений.

```
TTest1w_t ( weight, value)
```

ttest1w\_upper

Функция **TTest1w\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

```
TTest1w_upper (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:** число

**Аргументы:**

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 \(страница 344\)](#)

**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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 \(страница 344\)](#)

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

[р Создание типичного отчета t-test \(страница 344\)](#)

### TTest\_sig

Функция **TTest\_sig()** возвращает агрегированное значение двуххвостого 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.



## 5 Функции в скриптах и выражениях диаграммы

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

### Синтаксис:

```
TTest_sig (grp, value [, eq_var])
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:**

р *Создание типичного отчета t-test (страница 344)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 \(страница 344\)](#)

**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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 \(страница 344\)](#)

### 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.

## 5 Функции в скриптах и выражениях диаграммы

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:** число

### Аргументы:

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### TTestw\_dif

Функция **TTestw\_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

## 5 Функции в скриптах и выражениях диаграммы

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:** число

### Аргументы:

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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.

## 5 Функции в скриптах и выражениях диаграммы

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:** число

### Аргументы:

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р [Создание типичного отчета t-test \(страница 344\)](#)



### 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:** число

#### Аргументы:

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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: число

Аргументы:

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 )
```

## 5 Функции в скриптах и выражениях диаграммы

```
TTestw_sterr( weight, Group, value, false )
```

### См. также:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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 )
```

## 5 Функции в скриптах и выражениях диаграммы

`TTestw_t( weight, Group, value, false )`

**См. также:**

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

**Аргументы:**

Аргумент	Описание
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

#### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

#### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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:

р *Создание типичного отчета t-test (страница 344)*

### TTest1\_sig

Функция **TTest1\_sig()** возвращает агрегированное значение двуххвостого 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

**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:**

р *Создание типичного отчета t-test (страница 344)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .

### 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:**

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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:**

р *Создание типичного отчета t-test (страница 344)*

### TTest1w\_conf

Функция **TTest1w\_conf()** — это функция **numeric**, 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:** число

#### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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:

р [Создание типичного отчета t-test \(страница 344\)](#)

### 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:** число

#### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .

#### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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.

## 5 Функции в скриптах и выражениях диаграммы

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:** число

### Аргументы:

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .

### 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:

р *Создание типичного отчета t-test (страница 344)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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 \(страница 344\)](#)

### TTest1w\_sig

Функция **TTest1w\_sig()** возвращает агрегированное значение двуххвостого 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .

**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:**

р *Создание типичного отчета t-test (страница 344)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .

**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:**

р *Создание типичного отчета t-test (страница 344)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .

**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 (страница 344)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
weight	Каждое значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе <b>weight</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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 (страница 344)*

### 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.

*См.: Примеры использования функций z-test (страница 347)*

### Функции формата одного столбца

Следующие функции применяются к z-критериям с простыми сериями вводимых данных:

---

## 5 Функции в скриптах и выражениях диаграммы

---

### ztest\_conf

Функция **ZTest\_conf()** возвращает агрегированное z-значение для серии значений.

```
ZTest_conf (value [, sigma [, sig ]])
```

### ztest\_dif

Функция **ZTest\_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

```
ZTest_dif (value [, sigma])
```

### ztest\_sig

Функция **ZTest\_sig()** возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений.

```
ZTest_sig (value [, sigma])
```

### ztest\_sterr

Функция **ZTest\_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

```
ZTest_sterr (value [, sigma])
```

### ztest\_z

Функция **ZTest\_z()** возвращает агрегированное z-значение для серии значений.

```
ZTest_z (value [, sigma])
```

### ztest\_lower

Функция **ZTest\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

### ztest\_upper

Функция **ZTest\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
ZTest_upper (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.

```
ZTestw_conf (weight, value [, sigma [, sig]])
```

---

## 5 Функции в скриптах и выражениях диаграммы

---

ztestw\_dif

Функция **ZTestw\_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

```
ZTestw_dif (weight, value [, sigma])
```

ztestw\_lower

Функция **ZTestw\_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
ZTestw_lower (weight, value [, sigma])
```

ztestw\_sig

Функция **ZTestw\_sig()** возвращает агрегированное значение двуххвостого уровня важности z-критерия для серии значений.

```
ZTestw_sig (weight, value [, sigma])
```

ztestw\_sterr

Функция **ZTestw\_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

```
ZTestw_sterr (weight, value [, sigma])
```

ztestw\_upper

Функция **ZTestw\_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
ZTestw_upper (weight, value [, sigma])
```

ztestw\_z

Функция **ZTestw\_z()** возвращает агрегированное z-значение для серии значений.

```
ZTestw_z (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:** число

**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 <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**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 <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

**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:** число

**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 <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**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 <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**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 <b>sigma</b> . Если элемент <b>sigma</b> отсутствует, используется действительное стандартное отклонение выборки.
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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 (страница 347)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

*р Примеры использования функций z-test (страница 347)*

### 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.

## 5 Функции в скриптах и выражениях диаграммы

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

### Синтаксис:

```
ZTestw_z (weight, value [, sigma])
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
value	Эти значения возвращаются с помощью <b>value</b> . Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычитите это значение из выборки значений.
weight	Каждое выборочное значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению толщины в элементе <b>weight</b> .
sigma	Если стандартное отклонение известно, его можно указать в элементе <b>sigma</b> . Если элемент <b>sigma</b> 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:

р *Примеры использования функций z-test (страница 347)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Эти значения возвращаются с помощью <b>value</b> . 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 <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению толщины в элементе <b>weight</b> .
sigma	Если стандартное отклонение известно, его можно указать в элементе <b>sigma</b> . Если элемент <b>sigma</b> 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:

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Эти значения возвращаются с помощью <b>value</b> . Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычитите это значение из выборки значений.
weight	Каждое выборочное значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению толщины в элементе <b>weight</b> .
sigma	Если стандартное отклонение известно, его можно указать в элементе <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**Аргументы:**

Аргумент	Описание
value	Эти значения возвращаются с помощью <b>value</b> . Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычитите это значение из выборки значений.
weight	Каждое выборочное значение в элементе <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению толщины в элементе <b>weight</b> .
sigma	Если стандартное отклонение известно, его можно указать в элементе <b>sigma</b> . Если элемент <b>sigma</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

**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 <b>value</b> может подсчитываться один или несколько раз согласно соответствующему значению толщины в элементе <b>weight</b> .
sigma	Если стандартное отклонение известно, его можно указать в элементе <b>sigma</b> . Если элемент <b>sigma</b> отсутствует, используется действительное стандартное отклонение выборки.
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> 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:**

р *Примеры использования функций z-test (страница 347)*

**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:** число

**Аргументы:**

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:**

р *Примеры использования функций z-test (страница 347)*

### 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:** число

### Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе <b>group</b> . Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя <b>Value</b> .
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя <b>Type</b> .
sig	В <b>sig</b> можно указать двусторонний уровень важности. При отсутствии значения <b>sig</b> устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение <b>eq_var</b> определено как False (0), будут приняты отдельные изменения двух выборок. Если значение <b>eq_var</b> 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:

*р Примеры использования функций z-test (страница 347)*

## 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...
```

- Щелкните элемент **I** , чтобы загрузить данные.

### Создание визуализаций функции диаграммы chi2-test

#### Пример: Образец 1

Выполните следующие действия.

- В редакторе загрузки данных щелкните элемент ” , чтобы перейти в вид приложения, и нажмите ранее созданный лист.  
Откроется вид листа.
- Щелкните команду **@ Изменить**, чтобы изменить лист.
- Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте элементы Grp, Grade и Count в качестве измерений.  
В этой таблице показаны данные образца.
- 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.
- 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.
- 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))`

#### Result:

Полученная в результате таблица для функций chi2-test для данных образца 1 будет содержать следующие значения.

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).

#### Result:

Полученная в результате таблица для функций chi2-test для данных образца 2 будет содержать следующие значения.

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).

#### Result:

Полученная в результате таблица для функций chi2-test для данных образца 3 будет содержать следующие значения.

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.

## 5 Функции в скриптах и выражениях диаграммы

В этом примере используется таблица, содержащая количество студентов, достигших степени (A-F), для двух групп студентов (I и II).

	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 элемент I , чтобы загрузить данные.

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 элемент I , чтобы загрузить данные.

Значения chi2-test загружены в таблицу с именем Chi2\_table.

## 5 Функции в скриптах и выражениях диаграммы

### Результаты

Полученные значения chi2-test можно просмотреть в просмотре модели данных в разделе

**Предварительный просмотр**. Они должны выглядеть так:

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 Sense t-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

	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

### Загрузка данных образца

Выполните следующие действия.



1. Создайте новое приложение с новым листом и откройте этот лист.

2. Введите следующий редактор загрузки данных:

```
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 элемент **I**, чтобы загрузить данные.

### Создание таблицы Group Statistics

Выполните следующие действия.

1. В редакторе загрузки данных щелкните элемент **”**, чтобы перейти в вид приложения, и нажмите ранее созданный лист.  
Откроется вид листа.
2. Щелкните команду **@ Изменить**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте следующие выражения в качестве мер:

Метка	Выражение
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Добавьте элемент **Туре** в таблицу в качестве измерения.

## 5 Функции в скриптах и выражениях диаграммы

5. Нажмите кнопку **Сортировка** и переместите элемент Type в начало списка сортировки.

### Result:

Таблица Group Statistics для этих образцов будет выглядеть следующим образом:

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 команду **@ Изменить**, 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:

Метка	Выражение
conf	<code>if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))</code>
t	<code>if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))</code>
df	<code>if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))</code>
Sig. (2-tailed)	<code>if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))</code>
Mean Difference	<code>TTest_dif(Type, Value)</code>
Standard Error Difference	<code>if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))</code>
95% Confidence Interval of the Difference (Lower)	<code>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))</code>
95% Confidence Interval of the Difference (Upper)	<code>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))</code>

### Result:

Таблица **Independent Sample Test** для этих образцов будет выглядеть следующим образом:

	t	df	Sig. (2- taile d)	Mean Differenc e	Standard Error Differenc e	95% Confidenc e Interval of the Difference (Lower)	95% Confidenc e Interval of the Difference (Upper)
Equal Varianc e not Assume d	3.53	37.1167173358	0.001	15.2	4.30101	6.48625	23.9137
Equal Varianc e Assume d	3.53	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 элемент **I** , чтобы загрузить данные.

### Создание визуализаций функции диаграммы z-test

Выполните следующие действия.

1. В редакторе загрузки данных щелкните элемент ” , чтобы перейти в вид приложения, и нажмите лист, созданный при загрузке данных.  
Откроется вид листа.
2. Щелкните команду **@ Изменить**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте элемент **Туре** в качестве измерения.
4. Добавьте следующие выражения в таблицу в качестве мер.

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 форматирование числа мер, чтобы увидеть значимые значения. Таблицу будет легче считывать, если для большинства мер установить значение формата чисел **Номер>Простой** вместо **Auto**. Но, например, для элемента **ZTest Sig** используйте формат чисел: **Пользовательский**, а затем измените образец формата на **# ##**.

### Result:

Полученная в результате таблица для функций z-test для данных образца будет содержать следующие значения.

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:

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 текстовое значение, отсортированное в нескольких записях, как определено предложением **group by**.

**MaxString** (expression )

### MinString

Функция **MaxString()** находит строковые значения в выражении и возвращает первое текстовое значение, отсортированное в нескольких записях, как определено предложением **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 текстовое значение в порядке сортировки текста.

**MaxString** — функция диаграммы({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)

### MinString

Функция **MinString()** находит строковые значения в выражении или поле и возвращает первое текстовое значение в порядке сортировки текста.

**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.

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 <b>sort-weight</b> при его наличии со строкой, соответствующей наименьшему значению, появляющемуся в объединении первым.
distinct	Если слово <b>distinct</b> 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.

Пример	Результат	
<b>TeamData:</b> 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 ' ');  <b>Concat1:</b> LOAD SalesGroup,Concat(Team) as TeamConcat1 Resident TeamData Group By SalesGroup;	SalesGroup	TeamConcat1
	East	AlphaBetaDeltaGammaGamma
	West	EpsilonEtaThetaZeta
При условии, что таблица <b>TeamData</b> загружается, как в предыдущем примере:  LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;	SalesGroup	TeamConcat2
	East	Alpha-Beta-Delta-Gamma
	West	Epsilon-Eta-Theta-Zeta
При условии, что таблица <b>TeamData</b> загружается, как в предыдущем примере:  LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;	Поскольку аргумент для элемента <b>sort-weight</b> добавлен, порядок результатов определяется значением измерения Amount.	
	SalesGroup	TeamConcat2
	East	Delta-Beta-Gamma-Alpha
	West	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]])
```



## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** строка

**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 <b>sort-weight</b> 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 <b>DISTINCT</b> указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Примеры и результаты:**

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

## 5 Функции в скриптах и выражениях диаграммы

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 префикс DISTINCT означает что результат дубликата Gamma игнорируется. Также аргумент ограничителя определяется как запятая, после которой стоит пробел.
Concat (TOTAL <SalesGroup> Team)	Все значения строки для всех значений элемента Team связаны, если используется префикс TOTAL. Если указана выборка поля <SalesGroup>, результаты делятся на два значения измерения SalesGroup. Для элемента SalesGroup East результатами являются AlphaBetaDeltaGammaGamma. Для элемента SalesGroup West результатами являются EpsilonEtaThetaZeta.
Concat (TOTAL <SalesGroup> Team, '; ', Amount)	При добавлении аргумента для элемента <b>sort-weight</b> : 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: dual

### 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.

Пример	Результат	
<pre>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;</pre>	SalesGroup	FirstTeamLoaded
	East	Gamma
	West	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:** dual

**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	
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 текстовое значение, отсортированное в нескольких записях, как определено предложением **group by**.

**Синтаксис:**

```
MaxString ( expr )
```

**Возвращаемые типы данных:** dual

**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.

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 <b>TeamData</b> 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 текстовое значение в порядке сортировки текста.

**Синтаксис:**

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

**Возвращаемые типы данных:** dual

**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	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

**Ограничения:**

Если выражение не содержит значений со строковым представлением, возвращается значение NULL .

**Примеры и результаты:**

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

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
MaxString (Team)	Существует три значения 20000 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 <code>SET DateFormat='YYYY-MM-DD';</code> »

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 **MaxString()** находит строковые значения в выражении и возвращает первое текстовое значение, отсортированное в нескольких записях, как определено предложением **group by**.

**Синтаксис:**

```
MinString ( expr )
```

**Возвращаемые типы данных:** dual

**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.

## 5 Функции в скриптах и выражениях диаграммы

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	MinString1
	East	Alpha
	West	Epsilon
При условии, что таблица <b>TeamData</b> 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	MinString2
	East	01/05/2013
	West	01062/2013

### MinString — функция диаграммы

Функция **MinString()** находит строковые значения в выражении или поле и возвращает первое текстовое значение в порядке сортировки текста.

#### Синтаксис:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

**Возвращаемые типы данных:** dual

#### 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.



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
TOTAL	<p>Если слово <b>TOTAL</b> 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.</p> <p>При использовании выражения <b>TOTAL [&lt;fld {fld}&gt;]</b>, где префикс <b>TOTAL</b> предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.</p>

### Примеры и результаты:

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	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:** dual

**Аргументы:**

Аргумент	Описание
v1	Статическое значение (обычно выраженное строкой, но возможно и числом).
{,...}	Дополнительный список статических значений.

**Примеры и результаты:**

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: <table><tr><th colspan="4">ValueList()</th></tr><tr><th>Created dimension</th><th>Year</th><th colspan="2">Added expression</th></tr><tr><td></td><td></td><td></td><td>522.00</td></tr><tr><td>Number of Orders</td><td>2012</td><td></td><td>5.00</td></tr><tr><td>Number of Orders</td><td>2013</td><td></td><td>7.00</td></tr><tr><td>Average Order Size</td><td>2012</td><td></td><td>13.20</td></tr><tr><td>Average Order Size</td><td>2013</td><td></td><td>15.43</td></tr><tr><td>Total Amount</td><td>2012</td><td></td><td>66.00</td></tr><tr><td>Total Amount</td><td>2013</td><td></td><td>108.00</td></tr></table>	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
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 [
SalesID|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: dual

#### 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:

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.

Пример	Результат
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.

По общему правилу использование вложенных агрегирований в выражениях диаграмм программы Qlik Sense не допускается. Вложение допускается только в следующих случаях:

- При использовании префикса **TOTAL** во внутренней функции агрегирования.



Допустимо не более 100 уровней вложения.

### Вложенные агрегирования с префиксом 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))
```

Включение префикса **TOTAL** абсолютно необходимо для этого типа вложенности, допустимого программой Qlik Sense, но при этом необходимо для сравнения. Этот тип вложенности часто требуется и должен использоваться во всех подходящих случаях.

#### См. также:

р Aggr — функция диаграммы (страница 169)

## 5.2 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 **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()** используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется красным **r**, зеленым **g** и синим **b** компонентами со значениями от 0 до 255.

```
RGB(r, g, b)
```

### Color

Функция **Color()** используется в выражениях для возврата цветового представления цвета с номером **n** в палитре диаграммы, представленной в свойствах диаграммы. Представление цвета — это двойное значение, в котором текстовое представление указывается в виде 'RGB(r, g, b)', где **r**, **g** и **b** — числа от 0 до 255, представляющие значения красного, зеленого и синего цветов соответственно. Числовое представление — это целое число, представляющее красный, зеленый и синий компонент.

```
Color(n)
```

### 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  $0 < \text{Value} < 1$ , 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  $-1 < \text{Value} < 1$ , 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.

Функция цвета	Значение RGB
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	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:** dual

#### 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 коэффициент **alpha**. Следующие две позиции «00» обозначают количество **red**, следующие две позиции «FF» обозначают количество **green**, и последние две позиции «00» обозначают количество **blue**.

### RGB

**RGB()** используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется красным **r**, зеленым **g** и синим **b** компонентами со значениями от 0 до 255.

#### Синтаксис:

```
RGB (r, g, b)
```

**Возвращаемые типы данных:** dual

**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 коэффициент **alpha**. 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**.

### 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:** dual

**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).

### 5.3 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 (case1 [ , case2 , case3 , ... ] , else)
```

##### **class**

Функция **class** назначает первый параметр интервалу классов. Результат — двойное значение с уравнением  $a \leq x < b$  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 ]])
```

##### **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 число совпадающих выражений. При сравнении учитывается регистр.

```
match ( str, expr1 [ , expr2,...exprN ])
```

##### **mixmatch**

Функция **mixmatch** сравнивает первый параметр со всеми последующими и возвращает число 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. Она позволяет использовать знаки подстановки ( \* и ? ) в строках сравнения. 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:

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 и результаты:

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  $a \leq x < b$  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:

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 и результаты:

Пример	Результат
<code>class( var,10 ) c var = 23</code>	возвращает '20<=x<30'
<code>class( var,5,'value' ) c var = 23</code>	возвращает '20<= value <25'
<code>class( var,10,'x',5 ) c var = 23</code>	возвращает '15<=x<25'

#### Пример скрипта загрузки данных:

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. Исходная таблица выглядит следующим образом:

Name	Age
John	25
Karen	42
Yoshi	53

Чтобы добавить поле классификации по возрастной группе, можно добавить оператор предшествующей загрузки с помощью функции **class**. В этом примере мы загружаем исходную таблицу с помощью встроенных данных.

```
LOAD *,  
class(Age, 10, 'age') As Agegroup;
```

## 5 Функции в скриптах и выражениях диаграммы

```
LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

Полученные в результате загрузки данные выглядят так:

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

### if

Функция **if** возвращает значение в зависимости от условия функции: True или False.

#### Синтаксис:

```
if (condition , then , else)
```

Функция if имеет три параметра, *condition*, *then* и *else*, все из которых являются выражениями. Два остальных, *then* и *else*, могут относиться к любому типу.

#### Аргументы:

Аргумент	Описание
condition	Выражение, которое интерпретируется логическим образом.
then	Выражение, которое может быть любого типа. Если элемент <i>condition</i> равен True, функция if возвращает значение выражения <i>then</i> .
else	Выражение, которое может быть любого типа. Если элемент <i>condition</i> равен False, функция if возвращает значение выражения <i>else</i> .

Примеры и результаты:

Пример	Результат
if( Amount>= 0, 'OK', 'Alarm' )	Это выражение проверит, является ли количество положительным числом (0 или больше) и вернет значение 'OK', если это так. Если количество меньше 0, будет возвращено значение 'Alarm'.

### match

Функция **match** сравнивает первый параметр со всеми последующими и возвращает число 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**.

Примеры и результаты:

Пример	Результат
match( M, 'Jan','Feb','Mar')	возвращает 2, если M = Feb.  возвращает 0, если M = Apr или jan.

### mixmatch

Функция **mixmatch** сравнивает первый параметр со всеми последующими и возвращает число совпадающих выражений. При сравнении регистр не учитывается.

#### Синтаксис:

```
mixmatch( str, expr1 [ , expr2,...exprN ] )
```



Если 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**.

Примеры и результаты:

Пример	Результат
mixmatch( M, 'Jan','Feb','Mar')	возвращает 1, если M = jan

### pick

Функция отбора возвращает выражение *n* в списке.

#### Синтаксис:

```
pick(n, expr1[ , expr2,...exprN])
```

### Аргументы:

Аргумент	Описание
n	n представляет собой целое число от 1 до N.

### Примеры и результаты:

Пример	Результат
<code>pick( N, 'A','B',4, 6 )</code>	возвращает 'B', если N = 2 возвращает 4, если N = 3

## wildmatch

Функция **wildmatch** сравнивает первый параметр со всеми последующими и возвращает число совпадающих выражений. Она позволяет использовать знаки подстановки ( \* и ? ) в строках сравнения. При сравнении регистр не учитывается.

### Синтаксис:

```
wildmatch( str, expr1 [ , expr2,...exprN ] )
```



Если необходимо использовать сравнение без подстановочных знаков, используйте функции **match** или **mixmatch**.

### Примеры и результаты:

Пример	Результат
<code>wildmatch( M, 'ja*', 'fe?', 'mar' )</code>	возвращает 1, если M = January возвращает 2, если M = fex

## 5.4 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.



## 5 Функции в скриптах и выражениях диаграммы

---

### **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** выражением **while**. Первый шаг цикла — число 1. Функция **IterNo** имеет значение только при условии совместного использования с выражением **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 <b>autonumber</b> на различных ключах в скрипте, для названия каждого счетчика может использоваться дополнительный параметр <i>AutoID</i> .

#### 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.

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.

## 5 Функции в скриптах и выражениях диаграммы

```
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:

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
];
```

## 5 Функции в скриптах и выражениях диаграммы

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:

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

См. также:

р *autonumberhash128* (страница 380)

р *autonumberhash256* (страница 382)

### 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.

Region	Year	Month	Sales
North	2014	May	245

## 5 Функции в скриптах и выражениях диаграммы

Region	Year	Month	Sales
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:

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

В этом примере вы можете обратиться к RYMkey, например 1 вместо строки 'North2014May', если необходимо установить связь с другой таблицей.

Теперь мы загружаем исходную таблицу с ценами похожим образом. Поля Region, Year и Month исключены предыдущей загрузкой во избежание создания синтетического ключа, мы уже создаем составной ключ с функцией **autonumberhash128**, связывая таблицы.

## 5 Функции в скриптах и выражениях диаграммы

```
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:

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

**См. также:**

[p autonumberhash256 \(страница 382\)](#)

[p autonumber \(страница 378\)](#)

### 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.

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:

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2

## 5 Функции в скриптах и выражениях диаграммы

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 **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:

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

**См. также:**

р *autonumberhash128* (страница 380)  
р *autonumber* (страница 378)



### IterNo

Эта функция скрипта возвращает целое, указывающее на то, в который раз оценивается одна запись в операторе **LOAD** выражением **while**. Первый шаг цикла — число 1. Функция **IterNo** имеет значение только при условии совместного использования с выражением **while**.

#### Синтаксис:

```
IterNo ( )
```

#### Примеры и результаты:

Пример	Результат												
<pre>LOAD     IterNo() as Day,     Date( StartDate + IterNo() - 1 ) as Date     while StartDate + IterNo() - 1 &lt;= EndDate;</pre> <pre>LOAD * INLINE [StartDate, EndDate 2014-01-22, 2014-01-26 ];</pre>	<p>Данный оператор <b>LOAD</b> генерирует одну запись на дату внутри диапазона, определенного параметрами <b>StartDate</b> и <b>EndDate</b>.</p> <p>В этом примере полученная таблица будет выглядеть так:</p> <table><tr><th>Day</th><th>Date</th></tr><tr><td>1</td><td>2014-01-22</td></tr><tr><td>2</td><td>2014-01-23</td></tr><tr><td>3</td><td>2014-01-24</td></tr><tr><td>4</td><td>2014-01-25</td></tr><tr><td>5</td><td>2014-01-26</td></tr></table>	Day	Date	1	2014-01-22	2	2014-01-23	3	2014-01-24	4	2014-01-25	5	2014-01-26
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:

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:

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 межполевым порядке сортировки.



*Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используются функции **RowNo()**. Данные возможности сортировки автоматически отключаются.*

#### Синтаксис:

**RowNo** ( [ TOTAL ] )

**Возвращаемые типы данных:** целое число

#### Аргументы:

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.

#### Примеры и результаты:

Customer	UnitSales	Row in Segment	Row Number
Astrida	4	1	1
Astrida	10	2	2
Astrida	9	3	3
Betacab	5	1	4
Betacab	2	2	5
Betacab	25	3	6
Canutility	8	1	7
Canutility		2	8
Divadip	4	1	9
Divadip		2	10

Примеры	Результаты
Создайте визуализацию, состоящую из таблицы с измерениями <b>Customer</b> , <b>UnitSales</b> , и добавьте меры <code>rowNo( )</code> и <code>rowNo(TOTAL)</code> с метками <b>Row in Segment</b> и <b>Row Number</b> .	<p>Столбец <b>Row in Segment</b> показывает результаты 1, 2, 3 для сегмента столбца, содержащего значения поля <code>UnitSales</code> для клиента Astrida. Нумерация строк для следующего сегмента столбца, который является Betacab, начинается в таком случае снова с 1.</p> <p>Столбец <b>Row Number</b> игнорирует измерения, которые могут быть использованы при подсчете строк в таблице.</p>
Добавить выражение: <code>IF( rowNo( )=1, 0, UnitSales / Above( UnitSales ) )</code> как меру.	<p>Это выражение вернет значение 0 для первой строки в каждом сегменте столбца. Таким образом, в столбце будет показано следующее:</p> <p>0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2,375, 0 и 4.</p>

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 '|');
```

**См. также:**

[р Above](#) — функция диаграммы (страница 573)

### 5.5 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).

### 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)
```

## 5 Функции в скриптах и выражениях диаграммы

---

### month

Эта функция возвращает двойное значение с именем месяца, как определено переменной окружения **MonthNames**, и целое в диапазоне от 1 до 12. Месяц высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

```
month (expression)
```

### year

Эта функция возвращает год в виде целого числа, а выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

```
year (expression)
```

### weekyear

Эта функция возвращает год, которому принадлежит номер недели согласно стандарту ISO 8601. Номер недели в году может быть установлен в пределах от 1 до 52.

```
weekyear (expression)
```

### weekday

Эта функция возвращает двойное значение со следующим: Имя дня, как определено переменной окружения **DayNames**. Целое от 0 до 6, соответствующее номинальному дню недели (0–6).

```
weekday (date)
```

## Функции меток времени

### now

Эта функция возвращает метку текущего времени по системным часам. Значение по умолчанию — 1.

```
now ([ timer_mode])
```

### today

Эта функция возвращает текущую дату по системным часам.

```
today ([timer_mode])
```

### LocalTime

Эта функция возвращает метку текущего времени по системным часам для указанного часового пояса.

```
localtime ([timezone [, ignoreDST ]])
```

## Функции формирования

### makedate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, месяц **MM** и день **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

### **makeweekdate**

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, неделя **WW** и день недели **D**.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

### **maketime**

Эта функция возвращает время, рассчитанное в формате часы **hh**, минуты **mm** и секунды **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

## Другие функции даты

### **AddMonths**

Эта функция возвращает дату через **n** месяцев после даты начала **startdate** или, если **n** является отрицательным числом, — дату за **n** месяцев до даты начала **startdate**.

```
addmonths (startdate, n , [ , mode])
```

### **AddYears**

Эта функция возвращает дату через **n** лет после даты начала **startdate** или, если **n** является отрицательным числом, — дату за **n** лет до даты начала **startdate**.

```
addyears (startdate, n)
```

### **yeartodate**

Эта функция определяет, находится ли введенная метка времени в том году, в котором находится дата последней загрузки скрипта, и возвращает значение True, если это так, и False если это не так.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

## Функции часовых поясов

### **timezone**

Эта функция возвращает имя текущего часового пояса, соответствующее имени, используемому в Windows.

```
timezone ( )
```

### **GMT**

Эта функция возвращает текущее среднее время Greenwich Mean Time согласно системным часам и настройкам времени в Windows.

```
GMT ( )
```

### **UTC**

Возвращает текущее время Coordinated Universal Time.

```
UTC ( )
```



### daylightsaving

Возвращает текущие настройки перехода на летнее время согласно установкам Windows.

```
daylightsaving ( )
```

### converttolocaltime

Преобразует формат метки времени UTC или GMT в местное время и выводит в виде двойного значения. Местоположение может задаваться для любого числа городов, мест и часовых поясов Земли.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

## Функции установки времени

### setdateyear

Данная функция берет в качестве входных значений **timestamp** и **year** и обновляет значение **timestamp** с учетом указанного входного значения **year**.

```
setdateyear (timestamp, year)
```

### setdateyearmonth

Данная функция берет в качестве входных значений **timestamp**, **month** и **year** и обновляет значение **timestamp** с учетом указанных входных значений **year** и **month**.

```
setdateyearmonth (timestamp, year, month)
```

## Функции вхождения

### inyear

Эта функция возвращает значение True, если поле **timestamp** находится в пределах года, включающего значение, указанное в поле **base\_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

### inyeartodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части года, включающей значение, заданное в поле **base\_date** до последней миллисекунды, указанной в поле **base\_date**, включительно.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

### inquarter

Эта функция возвращает значение True, если поле **timestamp** находится в пределах квартала, включающего значение, указанное в поле **base\_date**.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

### inquartertodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части

## 5 Функции в скриптах и выражениях диаграммы

---

квартала, включающей значение, заданное в поле **base\_date** до последней миллисекунды, указанной в поле **base\_date**, включительно.

```
inquartertoday (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)
```

### **inmonthtoday**

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.

```
inmonthtoday (date, basedate , shift)
```

### **inmonths**

Эта функция определяет, находится ли метка времени в том же месяце, двухмесячном периоде, квартале, триместре или полугодии, что и базовая дата. Также можно проследить, находится ли эта метка времени в предыдущем или последующем временном периоде.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

### **inmonthstoday**

Эта функция определяет, находится ли метка времени в части месяца, двухмесячного периода, квартала, триместра или полугодия до последней миллисекунды, указанной в поле **base\_date**, включительно. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

```
inmonthstoday (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])
```

### **inweektoday**

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.

```
inweektoday (date, basedate , shift [, weekstart])
```

### **inlunarweek**

Эта функция определяет, находится ли значение **timestamp** в пределах лунной недели,

---

## 5 Функции в скриптах и выражениях диаграммы

---

включающей значение, указанное в поле **base\_date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
inlunarweek (date, basedate , shift [, weekstart])
```

### **inlunarweektoday**

Эта функция определяет, находится ли значение **timestamp** в пределах части лунной недели до последней миллисекунды, указанной в поле **base\_date**, включительно. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
inlunarweektoday (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]] )
```

## 5 Функции в скриптах и выражениях диаграммы

---

### quarterstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

### quarterend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

### quartername

Эта функция возвращает значение, отображающее месяцы квартала (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня квартала.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

### monthstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
monthstart (date [, shift = 0])
```

### monthend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
monthend (date [, shift = 0])
```

### monthname

Эта функция возвращает значение, отображающее месяц (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня указанного месяца.

```
monthname (date [, shift = 0])
```

### monthsstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

---

## 5 Функции в скриптах и выражениях диаграммы

---

### monthsend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

### monthsname

Эта функция возвращает значение, представляющее диапазон месяцев периода (форматированного согласно переменным скрипта **MonthNames**), а также года. Базовое числовое значение соответствует метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

### weekstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня (понедельника) календарной недели, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
weekstart (date [, shift = 0 [, weekoffset = 0]])
```

### weekend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последней даты (воскресенья) календарной недели, включающей дату, заданную в поле **date**. По умолчанию для вывода используется формат даты **DateFormat**, установленный в скрипте.

```
weekend (date [, shift = 0 [, weekoffset = 0]])
```

### weekname

Эта функция возвращает значение года и номер недели с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня недели, содержащего значение, указанное в поле **date**.

```
weekname (date [, shift = 0 [, weekoffset = 0]])
```

### lunarweekstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
lunarweekstart (date [, shift = 0 [, weekoffset = 0]])
```

### lunarweekend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

---

## 5 Функции в скриптах и выражениях диаграммы

---

```
lunarweekend (date [, shift = 0 [, weekoffset = 0]])
```

### **lunarweekname**

Эта функция возвращает значение года и номер лунной недели, соответствующие метке времени первой миллисекунды первого дня лунной недели, содержащего значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
lunarweekname (date [, shift = 0 [, weekoffset = 0]])
```

### **daystart**

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду дня, содержащуюся в аргументе **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

### **dayend**

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду дня, содержащуюся в поле **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

### **dayname**

Эта функция возвращает значение даты с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду дня, содержащего значение, указанное в поле **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

## Функции нумерации дней

### **age**

Функция **age** возвращает значение возраста в момент времени, заданный в поле **timestamp** (полных лет), человека, дата рождения которого указана в поле **date\_of\_birth**.

```
age (timestamp, date_of_birth)
```

### **networkdays**

Функция **networkdays** возвращает число рабочих дней (понедельник-пятница) между и включая значения, указанные в поле **start\_date** и **end\_date**, учитывая выходные, которые можно дополнительно задать в поле **holiday**.

```
networkdays (start:date, end_date {[, holiday]})
```

### **firstworkdate**

Функция **firstworkdate** возвращает самую позднюю дату начала, при которой период, заданный в поле **no\_of\_workdays** (понедельник-пятница), окончится не позднее даты, заданной в поле **end\_date**, с учетом возможных выходных. Параметры **end\_date** и **holiday** должны быть действительными

## 5 Функции в скриптах и выражениях диаграммы

датами или метками времени.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

### **lastworkdate**

Функция **lastworkdate** возвращает самую раннюю дату окончания для достижения указанного числа рабочих дней **no\_of\_workdays** (понедельник-пятница) с начальной датой **start\_date** и с учетом выходных, которые можно дополнительно задать в поле **holiday**. Поля **start\_date** и **holiday** должны быть действительными датами или метками времени.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

### **daynumberofyear**

Эта функция вычисляет номер дня года, на который приходится метка времени. Вычисление выполняется с первой миллисекунды первого дня года, но первый месяц может быть смещен.

```
daynumberofyear (date[,firstmonth])
```

### **daynumberofquarter**

Эта функция вычисляет номер дня квартала, на который приходится метка времени.

```
daynumberofquarter (date[,firstmonth])
```

## **addmonths**

Эта функция возвращает дату через **n** месяцев после даты начала **startdate** или, если **n** является отрицательным числом, — дату за **n** месяцев до даты начала **startdate**.

### **Синтаксис:**

```
AddMonths (startdate, n , [ , mode])
```

**Возвращаемые типы данных:** dual

### **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	Параметр <b>mode</b> указывает, добавляется ли месяц относительно начала или конца месяца. Если входная дата 28-го числа или выше, а параметр <b>mode</b> равен 1, то функция возвращает дату, которая отстоит на то же расстояние от конца месяца, что и входная дата. Значение mode по умолчанию — 0.

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

Пример	Результат
<code>addmonths ('2003-01-29',3)</code>	Возвращает «2003-04-29»
<code>addmonths ('2003-01-29',3,0)</code>	Возвращает «2003-04-29»
<code>addmonths ('2003-01-29',3,1)</code>	Возвращает «2003-04-28»
<code>addmonths ('2003-01-29',1,0)</code>	Возвращает «2003-02-28»
<code>addmonths ('2003-01-29',1,1)</code>	Возвращает «2003-02-26»
<code>addmonths ('2003-02-28',1,0)</code>	Возвращает «2003-03-28»
<code>addmonths ('2003-02-28',1,1)</code>	Возвращает «2003-03-31»

### 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:** dual

**Аргументы:**

Аргумент	Описание
<code>startdate</code>	Начальная дата в виде метки времени, например '2012-10-12'.
<code>n</code>	Количество лет в виде положительного или отрицательного целого числа.

Примеры и результаты:

Пример	Результат
<code>addyears ('2010-01-29',3)</code>	Возвращает «2013-01-29»
<code>addyears ('2010-01-29',-1)</code>	Возвращает «2009-01-29»

### age

Функция **age** возвращает значение возраста в момент времени, заданный в поле **timestamp** (полных лет), человека, дата рождения которого указана в поле **date\_of\_birth**.

**Синтаксис:**

```
age (timestamp, date_of_birth)
```

Может быть выражением.



**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Метка времени или выражение, определяемое по метке времени, до которой необходимо вычислить завершенное количество лет.
<b>date_of_birth</b>	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.

Пример	Результат																																							
age('25/01/2014', '29/10/2012')	Возвращает 1.																																							
age('29/10/2014', '29/10/2012')	Возвращает 2.																																							
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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;</pre>	<p>Результирующая таблица показывает возвращенные значения функции age для каждой записи в таблице.</p> <table><tr><th>Member</th><th>DateOfBirth</th><th>Age</th></tr><tr><td>John</td><td>28/03/1989</td><td>26</td></tr><tr><td>Linda</td><td>10/12/1990</td><td>24</td></tr><tr><td>Steve</td><td>5/2/1992</td><td>23</td></tr><tr><td>Birg</td><td>31/3/1993</td><td>22</td></tr><tr><td>Raj</td><td>19/5/1994</td><td>21</td></tr><tr><td>Prita</td><td>15/9/1994</td><td>20</td></tr><tr><td>Su</td><td>11/12/1994</td><td>20</td></tr><tr><td>Goran</td><td>2/3/1995</td><td>20</td></tr><tr><td>Sunny</td><td>14/5/1996</td><td>19</td></tr><tr><td>Ajoa</td><td>13/6/1996</td><td>19</td></tr><tr><td>Daphne</td><td>7/7/1998</td><td>17</td></tr><tr><td>Biffy</td><td>4/8/2000</td><td>15</td></tr></table>	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
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:** dual

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Метка времени или выражение, определяемое по метке времени, для преобразования.
<b>place</b>	<p>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.</p> <ul style="list-style-type: none"><li>• GMT</li><li>• GMT-12:00 - GMT-01:00</li><li>• GMT+01:00 - GMT+14:00</li><li>• UTC</li><li>• UTC-12:00 - UTC-01:00</li><li>• UTC+01:00 - UTC+14:00</li></ul> <div> Можно использовать только стандартные значения смещения времени. Невозможно использовать произвольное смещение времени, например GMT-04:27.</div>
<b>ignore_dst</b>	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

Abu Dhabi	Central America	Kabul	Newfoundland	Tashkent
-----------	-----------------	-------	--------------	----------

## 5 Функции в скриптах и выражениях диаграммы

### Действительные города и часовые пояса

Adelaide	Central Time (US & Canada)	Kamchatka	Novosibirsk	Tbilisi
Alaska	Chennai	Karachi	Nuku'alofa	Tehran
Amsterdam	Chihuahua	Kathmandu	Osaka	Tokyo
Arizona	Chongqing	Kolkata	Pacific Time (US & Canada)	Urumqi
Astana	Copenhagen	Krasnoyarsk	Paris	Warsaw
Athens	Darwin	Kuala Lumpur	Perth	Wellington
Atlantic Time (Canada)	Dhaka	Kuwait	Port Moresby	West Central Africa
Auckland	Eastern Time (US & Canada)	Kyiv	Prague	Vienna
Azores	Edinburgh	La Paz	Pretoria	Vilnius
Baghdad	Ekaterinburg	Lima	Quito	Vladivostok
Baku	Fiji	Lisbon	Riga	Volgograd
Bangkok	Georgetown	Ljubljana	Riyadh	Yakutsk
Beijing	Greenland	London	Rome	Yerevan
Belgrade	Greenwich Mean Time : Dublin	Madrid	Samoa	Zagreb
Berlin	Guadalajara	Magadan	Santiago	
Bern	Guam	Mazatlan	Sapporo	
Bogota	Hanoi	Melbourne	Sarajevo	
Brasilia	Harare	Mexico City	Saskatchewan	
Bratislava	Hawaii	Mid-Atlantic	Seoul	
Brisbane	Helsinki	Minsk	Singapore	
Brussels	Hobart	Monrovia	Skopje	
Bucharest	Hong Kong	Monterrey	Sofia	
Budapest	Indiana (East)	Moscow	Solomon Is.	

## 5 Функции в скриптах и выражениях диаграммы

### Действительные города и часовые пояса

Buenos Aires	International Date Line West	Mountain Time (US & Canada)	Sri Jayawardenepura
Cairo	Irkutsk	Mumbai	St. Petersburg
Canberra	Islamabad	Muscat	Stockholm
Cape Verde Is.	Istanbul	Nairobi	Sydney
Caracas	Jakarta	New Caledonia	Taipei
Casablanca	Jerusalem	New Delhi	Tallinn

Примеры и результаты:

Пример	Результат
<code>ConvertToLocalTime('2007-11-10 23:59:00', 'Paris')</code>	Возвращает '2007-11-11 00:59:00' и соответствующее внутреннее представление метки времени.
<code>ConvertToLocalTime(UTC(), 'GMT-05:00')</code>	Возвращает время для североамериканского восточного побережья, например Нью-Йорка.
<code>ConvertToLocalTime(UTC(), 'GMT-05:00', True)</code>	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.

#### Синтаксис:

**day** (expression)

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
<code>day( '1971-10-12' )</code>	возвращает 12
<code>day( '35648' )</code>	возвращает 6, так как 35648 = 1997-08-06

### dayend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду дня, содержащуюся в поле **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

#### Синтаксис:

```
DayEnd (time[, [period_no[, day_start]])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
<b>time</b>	Метка времени для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в поле <b>time</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие дни, положительные — последующие.
<b>day_start</b>	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре <b>day_start</b> . Например 0,125 обозначает 3:00 (3 AM).

#### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
dayend('25/01/2013 16:45:00')	Возвращает 25/01/2013 23:59:59.
dayend('25/01/2013 16:45:00', -1)	Возвращает 24/01/2013 23:59:59.
dayend('25/01/2013 16:45:00', 0, 0.5)	Возвращает 26/01/2013 11:59:59.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере обнаружена метка времени, которая отмечает окончание дня после даты каждого счета в таблице.</p> <p>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          ];</p> <p>InvoiceData:          LOAD *,          DayEnd(InvDate, 1) AS DEnd          Resident TempTable;          Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции dayend (). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>DEnd</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>29/03/2012 23:59:59</td></tr> <tr> <td>10/12/2012</td><td>11/12/2012 23:59:59</td></tr> <tr> <td>5/2/2013</td><td>07/02/2013 23:59:59</td></tr> <tr> <td>31/3/2013</td><td>01/04/2013 23:59:59</td></tr> <tr> <td>19/5/2013</td><td>20/05/2013 23:59:59</td></tr> <tr> <td>15/9/2013</td><td>16/09/2013 23:59:59</td></tr> <tr> <td>11/12/2013</td><td>12/12/2013 23:59:59</td></tr> <tr> <td>2/3/2014</td><td>03/03/2014 23:59:59</td></tr> <tr> <td>14/5/2014</td><td>15/05/2014 23:59:59</td></tr> <tr> <td>13/6/2014</td><td>14/06/2014 23:59:59</td></tr> <tr> <td>7/7/2014</td><td>08/07/2014 23:59:59</td></tr> <tr> <td>4/8/2014</td><td>05/08/2014 23:59:59</td></tr> </tbody> </table>	InvDate	DEnd	28/03/2012	29/03/2012 23:59:59	10/12/2012	11/12/2012 23:59:59	5/2/2013	07/02/2013 23:59:59	31/3/2013	01/04/2013 23:59:59	19/5/2013	20/05/2013 23:59:59	15/9/2013	16/09/2013 23:59:59	11/12/2013	12/12/2013 23:59:59	2/3/2014	03/03/2014 23:59:59	14/5/2014	15/05/2014 23:59:59	13/6/2014	14/06/2014 23:59:59	7/7/2014	08/07/2014 23:59:59	4/8/2014	05/08/2014 23:59:59
InvDate	DEnd																										
28/03/2012	29/03/2012 23:59:59																										
10/12/2012	11/12/2012 23:59:59																										
5/2/2013	07/02/2013 23:59:59																										
31/3/2013	01/04/2013 23:59:59																										
19/5/2013	20/05/2013 23:59:59																										
15/9/2013	16/09/2013 23:59:59																										
11/12/2013	12/12/2013 23:59:59																										
2/3/2014	03/03/2014 23:59:59																										
14/5/2014	15/05/2014 23:59:59																										
13/6/2014	14/06/2014 23:59:59																										
7/7/2014	08/07/2014 23:59:59																										
4/8/2014	05/08/2014 23:59:59																										

### daylightsaving

Возвращает текущие настройки перехода на летнее время согласно установкам Windows.

#### Синтаксис:

```
DaylightSaving( )
```

**Возвращаемые типы данных:** dual

#### Пример:

```
daylightsaving( )
```

### dayname

Эта функция возвращает значение даты с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду дня, содержащего значение, указанное в поле **time**.

### Синтаксис:

```
DayName (time[, period_no [, day_start]])
```

**Возвращаемые типы данных:** dual

### Аргументы:

Аргумент	Описание
<b>time</b>	Метка времени для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в полетime. Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие дни, положительные — последующие.
<b>day_start</b>	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре <b>day_start</b> . Например 0,125 обозначает 3:00 (3 AM).

### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

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.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере имя дня создано из метки времени, которая отмечает начало дня после даты каждого счета в таблице.</p> <pre>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;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции dayname(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table><tr><th>InvDate</th><th>DName</th></tr><tr><td>28/03/2012</td><td>29/03/2012 00:00:00</td></tr><tr><td>10/12/2012</td><td>11/12/2012 00:00:00</td></tr><tr><td>5/2/2013</td><td>07/02/2013 00:00:00</td></tr><tr><td>31/3/2013</td><td>01/04/2013 00:00:00</td></tr><tr><td>19/5/2013</td><td>20/05/2013 00:00:00</td></tr><tr><td>15/9/2013</td><td>16/09/2013 00:00:00</td></tr><tr><td>11/12/2013</td><td>12/12/2013 00:00:00</td></tr><tr><td>2/3/2014</td><td>03/03/2014 00:00:00</td></tr><tr><td>14/5/2014</td><td>15/05/2014 00:00:00</td></tr><tr><td>13/6/2014</td><td>14/06/2014 00:00:00</td></tr><tr><td>7/7/2014</td><td>08/07/2014 00:00:00</td></tr><tr><td>4/8/2014</td><td>05/08/2014 00:00:00</td></tr></table>	InvDate	DName	28/03/2012	29/03/2012 00:00:00	10/12/2012	11/12/2012 00:00:00	5/2/2013	07/02/2013 00:00:00	31/3/2013	01/04/2013 00:00:00	19/5/2013	20/05/2013 00:00:00	15/9/2013	16/09/2013 00:00:00	11/12/2013	12/12/2013 00:00:00	2/3/2014	03/03/2014 00:00:00	14/5/2014	15/05/2014 00:00:00	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
InvDate	DName																										
28/03/2012	29/03/2012 00:00:00																										
10/12/2012	11/12/2012 00:00:00																										
5/2/2013	07/02/2013 00:00:00																										
31/3/2013	01/04/2013 00:00:00																										
19/5/2013	20/05/2013 00:00:00																										
15/9/2013	16/09/2013 00:00:00																										
11/12/2013	12/12/2013 00:00:00																										
2/3/2014	03/03/2014 00:00:00																										
14/5/2014	15/05/2014 00:00:00																										
13/6/2014	14/06/2014 00:00:00																										
7/7/2014	08/07/2014 00:00:00																										
4/8/2014	05/08/2014 00:00:00																										

### daynumberofquarter

Эта функция вычисляет номер дня квартала, на который приходится метка времени.

#### Синтаксис:

```
DayNumberOfQuarter (timestamp[, start_month])
```

**Возвращаемые типы данных:** целое число

В этой функции год всегда включает 366 дней.

#### Аргументы:

Аргумент	Описание
timestamp	Дата для вычисления.



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>start_month</b>	Если в поле <b>start_month</b> 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 <b>start_month</b> = 3.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат																					
DayNumberOfQuarter('12/09/2014')	Возвращает 74, номер дня текущего квартала.																					
DayNumberOfQuarter('12/09/2014', 3)	Возвращает 12, номер дня текущего квартала. В этом случае первый квартал начинается с марта (поскольку элемент start_month указан как 3). Это означает, что текущий квартал является третьим кварталом, который начался первого сентября.																					
Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.  ProjectTable: LOAD recno() as Invid, * INLINE [ StartDate 28/03/2014 10/12/2014 5/2/2015 31/3/2015 19/5/2015 15/9/2015 ] ; NrDays: Load *, DayNumberOfQuarter(StartDate,4) As DayNrQtr Resident ProjectTable; Drop table ProjectTable;	Результирующая таблица показывает возвращенные значения функции DayNumberOfQuarter для каждой записи в таблице. <table><tr><th>Invid</th><th>StartDate</th><th>DayNrQtr</th></tr><tr><td>1</td><td>28/03/2014</td><td>88</td></tr><tr><td>2</td><td>10/12/2014</td><td>71</td></tr><tr><td>3</td><td>5/2/2015</td><td>36</td></tr><tr><td>4</td><td>31/3/2015</td><td>91</td></tr><tr><td>5</td><td>19/5/2015</td><td>49</td></tr><tr><td>6</td><td>15/9/2015</td><td>77</td></tr></table>	Invid	StartDate	DayNrQtr	1	28/03/2014	88	2	10/12/2014	71	3	5/2/2015	36	4	31/3/2015	91	5	19/5/2015	49	6	15/9/2015	77
Invid	StartDate	DayNrQtr																				
1	28/03/2014	88																				
2	10/12/2014	71																				
3	5/2/2015	36																				
4	31/3/2015	91																				
5	19/5/2015	49																				
6	15/9/2015	77																				

### daynumberofyear

Эта функция вычисляет номер дня года, на который приходится метка времени. Вычисление выполняется с первой миллисекунды первого дня года, но первый месяц может быть смещен.

### Синтаксис:

```
DayNumberOfYear (timestamp[,start_month])
```

**Возвращаемые типы данных:** целое число

В этой функции год всегда включает 366 дней.

### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата для вычисления.
<b>start_month</b>	Если в поле <b>start_month</b> 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 <b>start_month</b> = 3.

### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>DayNumberOfYear('12/09/2014')</code>	Возвращает 256, номер дня, отсчет которого начинается с первого дня года.
<code>DayNumberOfYear('12/09/2014', 3)</code>	Возвращает 196, номер дня, отсчет которого начинается с первого марта.

Пример	Результат																					
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>ProjectTable: LOAD recno() as InVID, * INLINE [ StartDate 28/03/2014 10/12/2014 5/2/2015 31/3/2015 19/5/2015 15/9/2015 ] ; NrDays: Load *, DayNumberOfYear(StartDate,4) As DayNrYear Resident ProjectTable; Drop table ProjectTable;</pre>	<p>Результирующая таблица показывает возвращенные значения функции DayNumberOfYear для каждой записи в таблице.</p> <table><tr><th>InVID</th><th>StartDate</th><th>DayNrYear</th></tr><tr><td>1</td><td>28/03/2014</td><td>363</td></tr><tr><td>2</td><td>10/12/2014</td><td>254</td></tr><tr><td>3</td><td>5/2/2015</td><td>311</td></tr><tr><td>4</td><td>31/3/2015</td><td>366</td></tr><tr><td>5</td><td>19/5/2015</td><td>49</td></tr><tr><td>6</td><td>15/9/2015</td><td>168</td></tr></table>	InVID	StartDate	DayNrYear	1	28/03/2014	363	2	10/12/2014	254	3	5/2/2015	311	4	31/3/2015	366	5	19/5/2015	49	6	15/9/2015	168
InVID	StartDate	DayNrYear																				
1	28/03/2014	363																				
2	10/12/2014	254																				
3	5/2/2015	311																				
4	31/3/2015	366																				
5	19/5/2015	49																				
6	15/9/2015	168																				

### daystart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду дня, содержащуюся в аргументе **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

#### Синтаксис:

```
DayStart(time[, [period_no[, day_start]])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
<b>time</b>	Метка времени для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в поле <b>time</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие дни, положительные — последующие.
<b>day_start</b>	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре <b>day_start</b> . Например 0,125 обозначает 3:00 (3 AM).

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат																										
<code>daystart('25/01/2013 16:45:00')</code>	Возвращает 25/01/2013 00:00:00.																										
<code>daystart('25/01/2013 16:45:00', -1)</code>	Возвращает 24/01/2013 00:00:00.																										
<code>daystart('25/01/2013 16:45:00', 0, 0.5 )</code>	Возвращает 25/01/2013 12:00:00.																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере обнаружена метка времени, которая отмечает начало дня после даты каждого счета в таблице.</p> <p>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 ];</p> <p>InvoiceData: LOAD *, DayStart(InvDate, 1) AS DStart Resident TempTable; Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>daystart()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table><thead><tr><th>InvDate</th><th>DStart</th></tr></thead><tbody><tr><td>28/03/2012</td><td>29/03/2012 00:00:00</td></tr><tr><td>10/12/2012</td><td>11/12/2012 00:00:00</td></tr><tr><td>5/2/2013</td><td>07/02/2013 00:00:00</td></tr><tr><td>31/3/2013</td><td>01/04/2013 00:00:00</td></tr><tr><td>19/5/2013</td><td>20/05/2013 00:00:00</td></tr><tr><td>15/9/2013</td><td>16/09/2013 00:00:00</td></tr><tr><td>11/12/2013</td><td>12/12/2013 00:00:00</td></tr><tr><td>2/3/2014</td><td>03/03/2014 00:00:00</td></tr><tr><td>14/5/2014</td><td>15/05/2014 00:00:00</td></tr><tr><td>13/6/2014</td><td>14/06/2014 00:00:00</td></tr><tr><td>7/7/2014</td><td>08/07/2014 00:00:00</td></tr><tr><td>4/8/2014</td><td>05/08/2014 00:00:00</td></tr></tbody></table>	InvDate	DStart	28/03/2012	29/03/2012 00:00:00	10/12/2012	11/12/2012 00:00:00	5/2/2013	07/02/2013 00:00:00	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
InvDate	DStart																										
28/03/2012	29/03/2012 00:00:00																										
10/12/2012	11/12/2012 00:00:00																										
5/2/2013	07/02/2013 00:00:00																										
31/3/2013	01/04/2013 00:00:00																										
19/5/2013	20/05/2013 00:00:00																										
15/9/2013	16/09/2013 00:00:00																										
11/12/2013	12/12/2013 00:00:00																										
2/3/2014	03/03/2014 00:00:00																										
14/5/2014	15/05/2014 00:00:00																										
13/6/2014	14/06/2014 00:00:00																										
7/7/2014	08/07/2014 00:00:00																										
4/8/2014	05/08/2014 00:00:00																										

### firstworkdate

Функция **firstworkdate** возвращает самую позднюю дату начала, при которой период, заданный в поле **no\_of\_workdays** (понедельник-пятница), окончится не позднее даты, заданной в поле **end\_date**, с учетом возможных выходных. Параметры **end\_date** и **holiday** должны быть действительными датами или метками времени.

## 5 Функции в скриптах и выражениях диаграммы

### Синтаксис:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

**Возвращаемые типы данных:** целое число

### Аргументы:

Аргумент	Описание
<b>end_date</b>	Метка времени даты окончания для оценки.
<b>no_of_workdays</b>	Количество рабочих дней, которое должно быть получено.
<b>holiday</b>	<p>Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми.</p> <p><b>Пример:</b> '25/12/2013', '26/12/2013'</p> <p>Можно указать несколько периодов выходных дней, разделенных запятыми.</p> <p><b>Пример:</b> '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'</p>

### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>firstworkdate ('29/12/2014', 9)</code>	Возвращает 17/12/2014.
<code>firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')</code>	Возвращает 15/12/2014, поскольку учитывается двухдневный период выходных.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																					
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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;</pre>	<p>Результирующая таблица показывает возвращенные значения функции FirstWorkDate для каждой записи в таблице.</p> <table><tr><th>InVID</th><th>EndDate</th><th>StartDate</th></tr><tr><td>1</td><td>28/03/2015</td><td>13/10/2014</td></tr><tr><td>2</td><td>10/12/2015</td><td>26/06/2015</td></tr><tr><td>3</td><td>5/2/2016</td><td>24/08/2015</td></tr><tr><td>4</td><td>31/3/2016</td><td>16/10/2015</td></tr><tr><td>5</td><td>19/5/2016</td><td>04/12/2015</td></tr><tr><td>6</td><td>15/9/2016</td><td>01/04/2016</td></tr></table>	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
InVID	EndDate	StartDate																				
1	28/03/2015	13/10/2014																				
2	10/12/2015	26/06/2015																				
3	5/2/2016	24/08/2015																				
4	31/3/2016	16/10/2015																				
5	19/5/2016	04/12/2015																				
6	15/9/2016	01/04/2016																				

### GMT

Эта функция возвращает текущее среднее время Greenwich Mean Time согласно системным часам и настройкам времени в Windows.

#### Синтаксис:

```
GMT ( )
```

**Возвращаемые типы данных:** dual

#### Пример:

```
gmt( )
```

### hour

Эта функция возвращает время в часах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

#### Синтаксис:

```
hour (expression)
```

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
hour( '09:14:36' )	возвращает 9
hour( '0.5555' )	возвращает 13 (так как 0,5555 = 13:19:55)

### inday

Эта функция возвращает значение True, если поле **timestamp** находится в пределах дня, включающего значение, указанное в поле **base\_timestamp**.

**Синтаксис:**

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

**Возвращаемые типы данных:** Boolean

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Дата и время, которые требуется сравнить с <b>base_timestamp</b> .
<b>base_timestamp</b>	Дата и время, используемые для оценки метки времени.
<b>period_no</b>	День можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает день, включающий значение, указанное в поле <b>base_timestamp</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие дни, положительные — последующие.
<b>day_start</b>	Если необходимо работать с днями, которые начинаются не в полночь, задайте смещение в виде десятичного значения в поле <b>day_start</b> , например, 0,125, чтобы день начинался в 3 часа (3 am).

Примеры и результаты:

Пример	Результат
inday ( '12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)	Возвращает True
inday ( '12/01/2006 12:23:00', '13/01/2006 00:00:00', 0)	Возвращает False
inday ( '12/01/2006 12:23:00', '12/01/2006 00:00:00', -1)	Возвращает False
inday ( '11/01/2006 12:23:00', '12/01/2006 00:00:00', -1)	Возвращает True

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<code>inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0, 0.5)</code>	Возвращает False																										
<code>inday ('12/01/2006 11:23:00', '12/01/2006 00:00:00', 0, 0.5)</code>	Возвращает True																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на период, начиная с <code>base_timestamp</code>.</p> <p>TempTable:  LOAD RecNo() as InvID, * Inline [    InvTime    28/03/2012    10/12/2012    5/2/2013    31/3/2013    19/5/2013    15/9/2013    11/12/2013    2/3/2014    14/5/2014    13/6/2014    7/7/2014    4/8/2014  ];</p> <p>InvoiceData:  LOAD *,  InDay(InvTime, '28/03/2012 00:00:00', 0) AS InDayEx  Resident TempTable;  Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inday()</code>.</p> <table> <thead> <tr> <th>InvTime</th><th>InDayEx</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>-1 (True)</td></tr> <tr> <td>10/12/2012</td><td>0 (False)</td></tr> <tr> <td>5/2/2013</td><td>0 (False)</td></tr> <tr> <td>31/3/2013</td><td>0 (False)</td></tr> <tr> <td>19/5/2013</td><td>0 (False)</td></tr> <tr> <td>15/9/2013</td><td>0 (False)</td></tr> <tr> <td>11/12/2013</td><td>0 (False)</td></tr> <tr> <td>2/3/2014</td><td>0 (False)</td></tr> <tr> <td>14/5/2014</td><td>0 (False)</td></tr> <tr> <td>13/6/2014</td><td>0 (False)</td></tr> <tr> <td>7/7/2014</td><td>0 (False)</td></tr> <tr> <td>4/8/2014</td><td>0 (False)</td></tr> </tbody> </table>	InvTime	InDayEx	28/03/2012	-1 (True)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvTime	InDayEx																										
28/03/2012	-1 (True)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										



### indaytotime

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части дня, включающей значение, заданное в поле **base\_timestamp** до определенной миллисекунды, указанной в поле **base\_timestamp**, включительно.

#### Синтаксис:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата и время, которые требуется сравнить с <b>base_timestamp</b> .
<b>base_timestamp</b>	Дата и время, используемые для оценки метки времени.
<b>period_no</b>	День можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает день, включающий значение, указанное в поле <b>base_timestamp</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие дни, положительные — последующие.
<b>day_start</b>	(дополнительно) Если необходимо работать с днями, которые начинаются не в полночь, задайте смещение в виде десятичного значения в поле <b>day_start</b> , например, 0,125, чтобы день начинался в 3 часа (3 am).

#### Примеры и результаты:

Пример	Результат
indaytotime ('12/01/2006 12:23:00', '12/01/2006 23:59:00', 0)	Возвращает True
indaytotime ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)	Возвращает False
indaytotime ('11/01/2006 12:23:00', '12/01/2006 23:59:00', -1)	Возвращает True

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли метка времени счета на период до 17:00:00 дня, начинающегося с base_timestamp.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvTime 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InDayToTime(InvTime, '28/03/2012 17:00:00', 0) AS InDayExTT Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции indaytotime().</p> <table> <thead> <tr> <th>InvTime</th><th>InDayExTT</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>-1 (True)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>0 (False)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>0 (False)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </tbody> </table>	InvTime	InDayExTT	28/03/2012	-1 (True)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvTime	InDayExTT																										
28/03/2012	-1 (True)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inlunarweek

Эта функция определяет, находится ли значение **timestamp** в пределах лунной недели, включающей значение, указанное в поле **base\_date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

#### Синтаксис:

```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки лунной недели.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>period_no</b>	Лунную неделю можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает лунную неделю, включающую значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие лунные недели, положительные — последующие.
<b>first_week_day</b>	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:

Пример	Результат
<code>inlunarweek('12/01/2013', '14/01/2013', 0)</code>	Возвращает True. Поскольку значение timestamp, 12/01/2013 выпадает на период с 08/01/2013 по 14/01/2013.
<code>inlunarweek('12/01/2013', '07/01/2013', 0)</code>	Возвращает False. Поскольку base_date 07/01/2013 находится в пределах лунной недели, определенной как период с 01/01/2013 по 07/01/2013.
<code>inlunarweek('12/01/2013', '14/01/2013', -1)</code>	Возвращает False. Поскольку при указании значения <b>period_no</b> как -1 происходит сдвиг недели на предыдущую неделю с 01/01/2013 по 07/01/2013.
<code>inlunarweek('07/01/2013', '14/01/2013', -1)</code>	Возвращает True. По сравнению с предыдущим примером метка времени находится в периоде с учетом сдвига назад.
<code>inlunarweek('11/01/2006', '08/01/2006', 0, 3)</code>	Возвращает False. Поскольку при указании значения для <b>first_week_day</b> как 3 начало года вычисляется от даты 04/01/2013, и, таким образом, значение <b>base_date</b> выпадает на первую неделю, а значение <b>timestamp</b> выпадает на период с 11/01/2013 по 17/01/2013.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на неделю, сдвинутую от значения <code>base_date</code> на четыре недели.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [   InvDate   28/03/2012   10/12/2012   5/2/2013   31/3/2013   19/5/2013   15/9/2013   11/12/2013   2/3/2014   14/5/2014   13/6/2014   7/7/2014   4/8/2014 ];  InvoiceData: LOAD *,   InLunarWeek(InvDate, '11/01/2013', 4)   AS InLWeekPlus4 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inlunarweek()</code>.</p> <p>Функция возвращает True для значения <code>InvDate</code> 5/2/2013, поскольку значение <code>base_date</code>, 11/01/2013, сдвигается на четыре недели, и, таким образом, выпадает на период с 5/02/2013 по 11/02/2013.</p> <table> <tr> <th>InvDate</th><th>InLWeekPlus4</th></tr> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>-1 (True)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>0 (False)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	InLWeekPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InLWeekPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inlunarweektodate

Эта функция определяет, находится ли значение **timestamp** в пределах части лунной недели до последней миллисекунды, указанной в поле **base\_date**, включительно. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

#### Синтаксис:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

**Возвращаемые типы данных:** Boolean

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки лунной недели.
<b>period_no</b>	Лунную неделю можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает лунную неделю, включающую значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие лунные недели, положительные — последующие.
<b>first_week_day</b>	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

**Примеры и результаты:**

Пример	Результат
<code>inlunarweektoday('12/01/2013', '13/01/2013', 0)</code>	Возвращает True. Поскольку значение <b>timestamp</b> , 12/01/2013, выпадает на период с 08/01/2013 по 13/01/2013.
<code>inlunarweektoday('12/01/2013', '11/01/2013', 0)</code>	Возвращает False. Поскольку значение <b>timestamp</b> имеет более позднюю дату, чем значение <b>base_date</b> , несмотря на то что обе даты приходятся на одну и ту же лунную неделю до 12/01/2012.
<code>inlunarweektoday('12/01/2006', '05/01/2006', 1)</code>	Возвращает True. При указании значения 1 для <b>period_no</b> происходит сдвиг <b>base_date</b> на одну неделю вперед, таким образом, значение <b>timestamp</b> выпадает на часть лунной недели.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на часть недели, сдвинутую от значения <code>base_date</code> на четыре недели.</p> <pre>TempTable: LOAD RecNo() as InVID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InLunarWeekToDate(InvDate, '07/01/2013', 4) AS InLWeek2DPlus4 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inlunarweek()</code>.</p> <p>Функция возвращает <code>True</code> для значения <code>InvDate</code> 5/2/2013, поскольку значение <code>base_date</code>, 11/01/2013, сдвигается на четыре недели, и, таким образом, выпадает на часть недели с 5/02/2013 по 07/02/2013.</p> <table><tr><th>InvDate</th><th>InLWeek2DPlus4</th></tr><tr><td>28/03/2012</td><td>0 (False)</td></tr><tr><td>10/12/2012</td><td>0 (False)</td></tr><tr><td>5/2/2013</td><td>-1 (True)</td></tr><tr><td>31/3/2013</td><td>0 (False)</td></tr><tr><td>19/5/2013</td><td>0 (False)</td></tr><tr><td>15/9/2013</td><td>0 (False)</td></tr><tr><td>11/12/2013</td><td>0 (False)</td></tr><tr><td>2/3/2014</td><td>0 (False)</td></tr><tr><td>14/5/2014</td><td>0 (False)</td></tr><tr><td>13/6/2014</td><td>0 (False)</td></tr><tr><td>7/7/2014</td><td>0 (False)</td></tr><tr><td>4/8/2014</td><td>0 (False)</td></tr></table>	InvDate	InLWeek2DPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InLWeek2DPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inmonth

Эта функция возвращает значение `True`, если поле **timestamp** находится в пределах месяца, включающего значение, указанное в поле **base\_date**.

#### Синтаксис:

```
InMonth (timestamp, base_date, period_no[, first_month_of_year])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>base_date</b>	Дата, используемая для оценки месяца.
<b>period_no</b>	Месяц можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает месяц, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие месяцы, положительные — последующие.
<b>first_month_of_year</b>	Параметр <b>first_month_of_year</b> отключен и зарезервирован для использования в будущем.

Примеры и результаты:

Пример	Результат																										
<code>inmonth ('25/01/2013', '01/01/2013', 0 )</code>	Возвращает True																										
<code>inmonth('25/01/2013', '01/04/2013', 0)</code>	Возвращает False																										
<code>inmonth ('25/01/2013', '01/01/2013', -1)</code>	Возвращает False																										
<code>inmonth ('25/12/2012', '01/01/2013', -1)</code>	Возвращает True																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на какой-нибудь день четвертого месяца после месяца в <b>base_date</b> при указании <b>period_no</b> как 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InMonth(InvDate, '31/01/2013', 4) AS InMthPlus4 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inmonth()</code>.</p> <table> <tr> <th>InvDate</th><th>InMthPlus4</th></tr> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>0 (False)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>-1 (True)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	InMthPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	-1 (True)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InMthPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	-1 (True)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inmonths

Эта функция определяет, находится ли метка времени в том же месяце, двухмесячном периоде, квартале, триместре или полугодии, что и базовая дата. Также можно проследить, находится ли эта метка времени в предыдущем или последующем временном периоде.

#### Синтаксис:

```
InMonths (n_months, timestamp, base_date, period_no [, first_month_of_year])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
<b>n_months</b>	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки периода.
<b>period_no</b>	Период можно сместить, задав значение в поле <b>period_no</b> , целое число или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие периоды, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

#### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>inmonths(4, '25/01/2013', '25/04/2013', 0)</code>	Возвращает True. Поскольку значение timestamp, 25/01/2013, находится в четырехмесячном периоде с 01/01/2013 по 30/04/2013, в котором находится значение base_date 25/04/2013.



## 5 Функции в скриптах и выражениях диаграммы

---

Пример	Результат
<code>inmonths(4, '25/05/2013', '25/04/2013', 0)</code>	Возвращает False. Поскольку 25/05/2013 находится за пределами периода, указанного в предыдущем примере.
<code>inmonths(4, '25/11/2012', '01/02/2013', -1 )</code>	Возвращает True. Поскольку значение <code>period_no</code> , -1, сдвигает период поиска на один период из четырех месяцев назад (значение <code>n-months</code> ), вследствие чего период поиска будет составлять промежуток с 01/09/2012 по 31/12/2012.
<code>inmonths( 4, '25/05/2006', '01/03/2006', 0, 3)</code>	Возвращает True. Поскольку для значения <code>first_month_of_year</code> задано значение 3, вследствие чего период поиска будет составлять промежуток с 01/03/2006 по 30/07/2006 вместо промежутка с 01/01/2006 по 30/04/2006.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета в таблице на двухмесячный период, который включает элемент <code>base_date</code>, смещенный вперед на один двухмесячный период (путем указания <code>period_no</code> как 1).</p> <p>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          ];</p> <p>InvoiceData:          LOAD *,          InMonths(2, InvDate, '11/02/2013', 1) AS InMthsPlus1          Resident TempTable;          Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>InMonths()</code>.</p> <p>Период поиска составляет промежуток с 01/03/2013 по 30/04/2013, поскольку значение <code>base_date</code> смещено на два месяца вперед от значения в функции (11/02/2013).</p> <table> <tr> <th>InvDate</th><th>InMthsPlus1</th></tr> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>0 (False)</td></tr> <tr><td>31/3/2013</td><td>-1 (True)</td></tr> <tr><td>19/5/2013</td><td>0 (False)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	InMthsPlus1	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	-1 (True)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InMthsPlus1																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	-1 (True)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inmonthstodate

Эта функция определяет, находится ли метка времени в части месяца, двухмесячного периода, квартала, триместра или полугодия до последней миллисекунды, указанной в поле **base\_date**, включительно. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

#### Синтаксис:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

## 5 Функции в скриптах и выражениях диаграммы

Возвращаемые типы данных: Boolean

Аргументы:

Аргумент	Описание
<b>n_months</b>	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки периода.
<b>period_no</b>	Период можно сместить, задав значение в поле <b>period_no</b> , целое число или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие периоды, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
inmonthstodate(4, '25/01/2013', '25/04/2013', 0)	Возвращает True. Поскольку значение timestamp, 25/01/2013, находится в четырехмесячном периоде с 01/01/2013 по конец 25/04/2013, в котором находится значение base_date 25/04/2013.
inmonthstodate(4, '26/04/2013', '25/04/2006', 0)	Возвращает False. Поскольку 26/04/2013 находится за пределами периода, указанного в предыдущем примере.
inmonthstodate(4, '25/09/2005', '01/02/2006', -1)	Возвращает True. Поскольку значение period_no, -1, сдвигает период поиска на один период из четырех месяцев назад (значение n-months), вследствие чего период поиска будет составлять промежуток с 01/09/2012 по 01/02/2012.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<code>inmonthstodate(4, '25/04/2006', '01/06/2006', 0, 3)</code>	Возвращает True. Поскольку для значения <code>first_month_of_year</code> задано значение 3, вследствие чего период поиска будет составлять промежуток с 01/03/2006 по 01/06/2006 вместо промежутка с 01/05/2006 по 01/06/2006.																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета в таблице на часть двухмесячного периода, который включает элемент <code>base_date</code>, смещенный вперед на четыре двухмесячных периода (путем указания <code>period_no</code> как 4).</p> <p>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 ];</p> <p>InvoiceData: LOAD *, InMonthsToDate(2, InvDate, '15/02/2013', 4) AS InMths2DPlus4 Resident TempTable; Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>InMonths()</code>.</p> <p>Период поиска составляет промежуток с 01/09/2013 по 15/10/2013, поскольку значение <code>base_date</code> смещено на восемь месяцев вперед от значения в функции (15/02/2013).</p> <table><thead><tr><th>InvDate</th><th>InMths2DPlus4</th></tr></thead><tbody><tr><td>28/03/2012</td><td>0 (False)</td></tr><tr><td>10/12/2012</td><td>0 (False)</td></tr><tr><td>5/2/2013</td><td>0 (False)</td></tr><tr><td>31/3/2013</td><td>0 (False)</td></tr><tr><td>19/5/2013</td><td>0 (False)</td></tr><tr><td>15/9/2013</td><td>-1 (True)</td></tr><tr><td>11/12/2013</td><td>0 (False)</td></tr><tr><td>2/3/2014</td><td>0 (False)</td></tr><tr><td>14/5/2014</td><td>0 (False)</td></tr><tr><td>13/6/2014</td><td>0 (False)</td></tr><tr><td>7/7/2014</td><td>0 (False)</td></tr><tr><td>4/8/2014</td><td>0 (False)</td></tr></tbody></table>	InvDate	InMths2DPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	-1 (True)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InMths2DPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	-1 (True)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inmonthtodate

Возвращает значение True, если значение **date** находится в пределах части месяца, включающей значение, заданное в поле **basedate** до последней миллисекунды, указанной в поле **basedate**, включительно.

#### Синтаксис:

```
InMonthToDate (timestamp, base_date, period_no)
```

**Возвращаемые типы данных:** Boolean

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки месяца.
<b>period_no</b>	Месяц можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает месяц, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие месяцы, положительные — последующие.

**Примеры и результаты:**

Пример	Результат
<code>inmonthtodate ('25/01/2013', '25/01/2013', 0)</code>	Возвращает True
<code>inmonthtodate ('25/01/2013', '24/01/2013', 0)</code>	Возвращает False
<code>inmonthtodate ('25/01/2013', '28/02/2013', -1)</code>	Возвращает True

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>При указании значения 4 для <code>period_no</code> в этом примере проверяется, выпадает ли дата счета на четвертый месяц после месяца в <code>base_date</code>, но до конца дня, указанного в <code>base_date</code>.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InMonthToDate(InvDate, '31/01/2013', 0, 4) AS InMthPlus42D Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inmonthtoday()</code>.</p> <table> <thead> <tr> <th>InvDate</th><th>InMthPlus42D</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>0 (False)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>-1 (True)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </tbody> </table>	InvDate	InMthPlus42D	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	-1 (True)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InMthPlus42D																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	-1 (True)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### 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:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки квартала.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>period_no</b>	Квартал можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает квартал, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие кварталы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

Пример	Результат
<code>inquarter ('25/01/2013', '01/01/2013', 0)</code>	Возвращает True
<code>inquarter ('25/01/2013', '01/04/2013', 0)</code>	Возвращает False
<code>inquarter ('25/01/2013', '01/01/2013', -1)</code>	Возвращает False
<code>inquarter ('25/12/2012', '01/01/2013', -1)</code>	Возвращает True
<code>inquarter ('25/01/2013', '01/03/2013', 0, 3)</code>	Возвращает False
<code>inquarter ('25/03/2013', '01/03/2013', 0, 3)</code>	Возвращает True

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на четвертый квартал финансового года, указанного путем указания значения 4 для элемента <code>first_month_of_year</code>, при этом значение для <code>base_date</code> равно 31/01/2013.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InQuarter(InvDate, '31/01/2013', 0, 4) AS Qtr4FinYr1213 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inquarter()</code>.</p> <table> <tr> <th>InvDate</th><th>Qtr4Fin1213</th></tr> <tr> <td>28/03/2012</td><td>0 (False)</td></tr> <tr> <td>10/12/2012</td><td>0 (False)</td></tr> <tr> <td>5/2/2013</td><td>-1 (True)</td></tr> <tr> <td>31/3/2013</td><td>-1 (True)</td></tr> <tr> <td>19/5/2013</td><td>0 (False)</td></tr> <tr> <td>15/9/2013</td><td>0 (False)</td></tr> <tr> <td>11/12/2013</td><td>0 (False)</td></tr> <tr> <td>2/3/2014</td><td>0 (False)</td></tr> <tr> <td>14/5/2014</td><td>0 (False)</td></tr> <tr> <td>13/6/2014</td><td>0 (False)</td></tr> <tr> <td>7/7/2014</td><td>0 (False)</td></tr> <tr> <td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	Qtr4Fin1213	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	-1 (True)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	Qtr4Fin1213																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	-1 (True)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### 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:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>base_date</b>	Дата, использующаяся для оценки квартала.
<b>period_no</b>	Квартал можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает квартал, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие кварталы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

Пример	Результат																										
<code>inquartertodate ('25/01/2013', '25/01/2013', 0)</code>	Возвращает True																										
<code>inquartertodate ('25/01/2013', '24/01/2013', 0)</code>	Возвращает False																										
<code>inquartertodate ('25/01/2012', '01/02/2013', -1)</code>	Возвращает True																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента <code>first_month_of_year</code>, и на четвертую четверть до окончания 28/02/2013.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InQuarterToDate(InvDate, '28/02/2013', 0, 4) AS Qtr42Date Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inquartertodate()</code>.</p> <table> <tr> <th>InvDate</th><th>Qtr42Date</th></tr> <tr> <td>28/03/2012</td><td>0 (False)</td></tr> <tr> <td>10/12/2012</td><td>0 (False)</td></tr> <tr> <td>5/2/2013</td><td>-1 (True)</td></tr> <tr> <td>31/3/2013</td><td>0 (False)</td></tr> <tr> <td>19/5/2013</td><td>0 (False)</td></tr> <tr> <td>15/9/2013</td><td>0 (False)</td></tr> <tr> <td>11/12/2013</td><td>0 (False)</td></tr> <tr> <td>2/3/2014</td><td>0 (False)</td></tr> <tr> <td>14/5/2014</td><td>0 (False)</td></tr> <tr> <td>13/6/2014</td><td>0 (False)</td></tr> <tr> <td>7/7/2014</td><td>0 (False)</td></tr> <tr> <td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	Qtr42Date	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	Qtr42Date																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### 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:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки недели.
<b>period_no</b>	Неделю можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает неделю, включающую значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие недели, положительные — последующие.
<b>first_week_day</b>	По умолчанию первым днем недели является понедельник, началом которого является полночь в ночь с воскресенья на понедельник. Чтобы задать другой день в качестве начала недели, укажите смещение в поле <b>first_week_day</b> . Это может быть целое число дней и/или десятичное значение.

#### Примеры и результаты:

Пример	Результат
<code>inweek ('12/01/2006', '14/01/2006', 0)</code>	Возвращает True
<code>inweek ('12/01/2006', '20/01/2006', 0 )</code>	Возвращает False
<code>inweek ('12/01/2006', '14/01/2006', -1 )</code>	Возвращает False
<code>inweek ('07/01/2006', '14/01/2006', -1)</code>	Возвращает True
<code>inweek ('12/01/2006', '09/01/2006', 0, 3)</code>	Возвращает False , поскольку для элемента <code>first_week_day</code> указано значение 3 (четверг), в результате чего элемент 12/01/2006 становится первым днем недели после недели с элементом 09/01/2006.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на какой-нибудь день четвертой недели после недели в <code>base_date</code> при указании значения 4 для элемента <code>period_no</code>.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [   InvDate   28/03/2012   10/12/2012   5/2/2013   31/3/2013   19/5/2013   15/9/2013   11/12/2013   2/3/2014   14/5/2014   13/6/2014   7/7/2014   4/8/2014 ];  InvoiceData: LOAD *,   InWeek(InvDate, '11/01/2013', 4) AS InWeekPlus4 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inweek()</code>.</p> <p>InvDate5/2/2013 выпадает на неделю через четыре недели после <code>base_date</code>: 11/1/2013.</p> <table><tr><th>InvDate</th><th>InWeekPlus4</th></tr><tr><td>28/03/2012</td><td>0 (False)</td></tr><tr><td>10/12/2012</td><td>0 (False)</td></tr><tr><td>5/2/2013</td><td>-1 (True)</td></tr><tr><td>31/3/2013</td><td>0 (False)</td></tr><tr><td>19/5/2013</td><td>0 (False)</td></tr><tr><td>15/9/2013</td><td>0 (False)</td></tr><tr><td>11/12/2013</td><td>0 (False)</td></tr><tr><td>2/3/2014</td><td>0 (False)</td></tr><tr><td>14/5/2014</td><td>0 (False)</td></tr><tr><td>13/6/2014</td><td>0 (False)</td></tr><tr><td>7/7/2014</td><td>0 (False)</td></tr><tr><td>4/8/2014</td><td>0 (False)</td></tr></table>	InvDate	InWeekPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InWeekPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inweektoday

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:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>base_date</b>	Дата, используемая для оценки недели.
<b>period_no</b>	Неделю можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает неделю, включающую значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие недели, положительные — последующие.
<b>first_week_day</b>	По умолчанию первым днем недели является понедельник, началом которого является полночь в ночь с воскресенья на понедельник. Чтобы задать другой день в качестве начала недели, укажите смещение в поле <b>first_week_day</b> . Это может быть целое число дней и/или десятичное значение.

Примеры и результаты:

Пример	Результат
<code>inweektoday ('12/01/2006', '12/01/2006', 0)</code>	Возвращает True
<code>inweektoday ('12/01/2006', '11/01/2006', 0)</code>	Возвращает False
<code>inweektoday ('12/01/2006', '18/01/2006', -1)</code>	Возвращает False , поскольку для элемента <b>period_no</b> указано значение -1, при этом дата вступления в силу <b>timestamp</b> измеряется на основе 11/01/2006.
<code>inweektoday ('11/01/2006', '12/01/2006', 0, 3)</code>	Возвращает False , поскольку для элемента <b>first_week_day</b> указано значение 3 (четверг), в результате чего элемент 12/01/2006 становится первым днем недели после недели с элементом 12/01/2006.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на четвертую неделю после недели в <code>base_date</code> путем указания значения 4 для элемента <code>period_no</code>, но до значения <code>base_date</code>.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InWeekToDate(InvDate, '11/01/2013', 4) AS InWeek2DPlus4 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inweek()</code>.</p> <table> <tr> <th>InvDate</th><th>InWeek2DPlus4</th></tr> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>0 (False)</td></tr> <tr><td>5/2/2013</td><td>-1 (True)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>0 (False)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	InWeek2DPlus4	28/03/2012	0 (False)	10/12/2012	0 (False)	5/2/2013	-1 (True)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	InWeek2DPlus4																										
28/03/2012	0 (False)																										
10/12/2012	0 (False)																										
5/2/2013	-1 (True)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inyear

Эта функция возвращает значение `True`, если поле **timestamp** находится в пределах года, включающего значение, указанное в поле **base\_date**.

#### Синтаксис:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>base_date</b>	Дата, используемая для оценки года.
<b>period_no</b>	Год можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает год, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие годы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>inyear ('25/01/2013', '01/01/2013', 0 )</code>	Возвращает True
<code>inyear ('25/01/2012', '01/01/2013', 0)</code>	Возвращает False
<code>inyear ('25/01/2013', '01/01/2013', -1)</code>	Возвращает False
<code>inyear ('25/01/2012', '01/01/2013', -1 )</code>	Возвращает True
<code>inyear ('25/01/2013', '01/01/2013', 0, 3)</code>	Возвращает False Значение <b>base_date</b> и <b>first_month_of_year</b> указывает, что значение <b>timestamp</b> должно выпадать на период с 01/03/2012 по 28/02/2013
<code>inyear ('25/03/2013', '2013/07/01', 0, 3 )</code>	Возвращает True

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента <code>first_month_of_year</code>, при этом значение для <code>base_date</code> составляет промежуток с 1/4/2012 по 31/03/2013.</p> <pre>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 ];</pre> <p>Проверьте, выпадает ли значение <code>InvDate</code> на финансовый год в промежутке с 1/04/2012 по 31/03/2013:</p> <pre>InvoiceData: LOAD *, InYear(InvDate, '31/01/2013', 0, 4) AS FinYr1213 Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inyear()</code>.</p> <table> <tr> <td>InvDate</td><td>FinYr1213</td></tr> <tr> <td>28/03/2012</td><td>0 (False)</td></tr> <tr> <td>10/12/2012</td><td>-1 (True)</td></tr> <tr> <td>5/2/2013</td><td>-1 (True)</td></tr> <tr> <td>31/3/2013</td><td>-1 (True)</td></tr> <tr> <td>19/5/2013</td><td>0 (False)</td></tr> <tr> <td>15/9/2013</td><td>0 (False)</td></tr> <tr> <td>11/12/2013</td><td>0 (False)</td></tr> <tr> <td>2/3/2014</td><td>0 (False)</td></tr> <tr> <td>14/5/2014</td><td>0 (False)</td></tr> <tr> <td>13/6/2014</td><td>0 (False)</td></tr> <tr> <td>7/7/2014</td><td>0 (False)</td></tr> <tr> <td>4/8/2014</td><td>0 (False)</td></tr> </table>	InvDate	FinYr1213	28/03/2012	0 (False)	10/12/2012	-1 (True)	5/2/2013	-1 (True)	31/3/2013	-1 (True)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	FinYr1213																										
28/03/2012	0 (False)																										
10/12/2012	-1 (True)																										
5/2/2013	-1 (True)																										
31/3/2013	-1 (True)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### inyeartodate

Эта функция возвращает значение `True`, если значение **timestamp** находится в пределах части года, включающей значение, заданное в поле **base\_date** до последней миллисекунды, указанной в поле **base\_date**, включительно.

#### Синтаксис:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

**Возвращаемые типы данных:** Boolean

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Дата, которую необходимо сравнить со значением, указанным в поле <b>base_date</b> .
<b>base_date</b>	Дата, используемая для оценки года.
<b>period_no</b>	Год можно сместить, задав значение в поле <b>period_no</b> . <b>period_no</b> — целое число, где 0 обозначает год, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие годы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

**Примеры и результаты:**

Пример	Результат
<code>inyeartodate ('2013/01/25', '2013/02/01', 0)</code>	Возвращает True
<code>inyeartodate ('2012/01/25', '2013/01/01', 0)</code>	Возвращает False
<code>inyeartodate ('2012/01/25', '2013/02/01', - )</code>	Возвращает True
<code>inyeartodate ('2012/11/25', '2013/01/31', 0, 4)</code>	Возвращает True Значение timestamp выпадает на финансовый год, начинающийся в четвертом месяце, и до значения base_date.
<code>inyeartodate ( '2013/3/31', '2013/01/31', 0, 4 )</code>	Возвращает False По сравнению с предыдущим примером, значение timestamp все еще находится в фискальном году, но после значения base_date, таким образом, оно находится за пределами части года.



Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента <code>first_month_of_year</code>, и на часть года до окончания 31/01/2013.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, InYearToDate(InvDate, '31/01/2013', 0, 4) AS FinYr2Date Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>inyeartodate()</code>.</p> <table> <thead> <tr> <th>InvDate</th><th>FinYr2Date</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>0 (False)</td></tr> <tr><td>10/12/2012</td><td>-1 (True)</td></tr> <tr><td>5/2/2013</td><td>0 (False)</td></tr> <tr><td>31/3/2013</td><td>0 (False)</td></tr> <tr><td>19/5/2013</td><td>0 (False)</td></tr> <tr><td>15/9/2013</td><td>0 (False)</td></tr> <tr><td>11/12/2013</td><td>0 (False)</td></tr> <tr><td>2/3/2014</td><td>0 (False)</td></tr> <tr><td>14/5/2014</td><td>0 (False)</td></tr> <tr><td>13/6/2014</td><td>0 (False)</td></tr> <tr><td>7/7/2014</td><td>0 (False)</td></tr> <tr><td>4/8/2014</td><td>0 (False)</td></tr> </tbody> </table>	InvDate	FinYr2Date	28/03/2012	0 (False)	10/12/2012	-1 (True)	5/2/2013	0 (False)	31/3/2013	0 (False)	19/5/2013	0 (False)	15/9/2013	0 (False)	11/12/2013	0 (False)	2/3/2014	0 (False)	14/5/2014	0 (False)	13/6/2014	0 (False)	7/7/2014	0 (False)	4/8/2014	0 (False)
InvDate	FinYr2Date																										
28/03/2012	0 (False)																										
10/12/2012	-1 (True)																										
5/2/2013	0 (False)																										
31/3/2013	0 (False)																										
19/5/2013	0 (False)																										
15/9/2013	0 (False)																										
11/12/2013	0 (False)																										
2/3/2014	0 (False)																										
14/5/2014	0 (False)																										
13/6/2014	0 (False)																										
7/7/2014	0 (False)																										
4/8/2014	0 (False)																										

### lastworkdate

Функция **lastworkdate** возвращает самую раннюю дату окончания для достижения указанного числа рабочих дней **no\_of\_workdays** (понедельник-пятница) с начальной датой **start\_date** и с учетом выходных, которые можно дополнительно задать в поле **holiday**. Поля **start\_date** и **holiday** должны быть действительными датами или метками времени.

#### Синтаксис:

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>start_date</b>	Начальная дата для вычисления.
<b>no_of_workdays</b>	Количество рабочих дней, которое должно быть получено.
<b>holiday</b>	<p>Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми.</p> <p><b>Пример:</b> '25/12/2013', '26/12/2013'</p> <p>Можно указать несколько периодов выходных дней, разделенных запятыми.</p> <p><b>Пример:</b> '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'</p>

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
lastworkdate ('19/12/2014', 9)	Возвращает «31/12/2014»
lastworkdate ('19/12/2014', 9, '2014-12-25', '2014-12-26')	Возвращает «02/01/2015», поскольку учитывается двухдневный период выходных.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																					
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>ProjectTable: LOAD *, recno() as InvID, INLINE [ StartDate 28/03/2014 10/12/2014 5/2/2015 31/3/2015 19/5/2015 15/9/2015 ] ; NrDays: Load *, LastWorkDate(StartDate,120) As EndDate Resident ProjectTable; Drop table ProjectTable;</pre>	<p>Результирующая таблица показывает возвращенные значения функции LastWorkDate для каждой записи в таблице.</p> <table><tr><th>InvID</th><th>StartDate</th><th>EndDate</th></tr><tr><td>1</td><td>28/03/2014</td><td>11/09/2014</td></tr><tr><td>2</td><td>10/12/2014</td><td>26/05/2015</td></tr><tr><td>3</td><td>5/2/2015</td><td>27/07/2015</td></tr><tr><td>4</td><td>31/3/2015</td><td>14/09/2015</td></tr><tr><td>5</td><td>19/5/2015</td><td>02/11/2015</td></tr><tr><td>6</td><td>15/9/2015</td><td>29/02/2016</td></tr></table>	InvID	StartDate	EndDate	1	28/03/2014	11/09/2014	2	10/12/2014	26/05/2015	3	5/2/2015	27/07/2015	4	31/3/2015	14/09/2015	5	19/5/2015	02/11/2015	6	15/9/2015	29/02/2016
InvID	StartDate	EndDate																				
1	28/03/2014	11/09/2014																				
2	10/12/2014	26/05/2015																				
3	5/2/2015	27/07/2015																				
4	31/3/2015	14/09/2015																				
5	19/5/2015	02/11/2015																				
6	15/9/2015	29/02/2016																				

### localtime

Эта функция возвращает метку текущего времени по системным часам для указанного часового пояса.

#### Синтаксис:

```
LocalTime ([timezone [, ignoreDST ]])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
timezone	<p>Параметр <b>timezone</b> задается как строка, содержащая любое географическое название, указанное в разделе <b>Time Zone</b> в <b>Windows Control Panel</b> для поля <b>Date and Time</b> или в виде строки в формате 'GMT+чч:мм'.</p> <p>Если часовой пояс не задан, будет возвращено местное время.</p>
ignoreDST	Если элемент <b>ignoreDST</b> 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.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат
<code>localtime ()</code>	Возвращает местное время в виде 2014–10–22 12:54:47.
<code>localtime ('London')</code>	Возвращает местное время в Лондоне в виде 2014–10–22 11:54:47.
<code>localtime ('GMT+02:00')</code>	Возвращает местное время в часовом поясе GMT+02:00, 2014–10–22 13:54:47.
<code>localtime ('Paris', -1')</code>	Возвращает местное время в Париже без учета перехода на летнее время 2014–10–22 11:54:47.

### lunarweekend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

#### Синтаксис:

```
LunarweekEnd (date[, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие лунные недели, положительные — последующие.
<b>first_week_day</b>	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

#### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>lunarweekend('12/01/2013')</code>	Возвращает 14/01/2013 23:59:59.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<code>lunarweekend('12/01/2013', -1)</code>	Возвращает 7/01/2013 23:59:59.																										
<code>lunarweekend('12/01/2013', 0, 1)</code>	Возвращает 15/01/2013 23:59:59.																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение последнего дня лунной недели каждой даты счета в таблице, где значение date смещается на одну неделю путем указания для элемента period_no значения 1.</p> <p>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 ];</p> <p>InvoiceData: LOAD *, LunarWeekEnd(InvDate, 1) AS LwkEnd Resident TempTable; Drop table TempTable;</p>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции lunarweekend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table><thead><tr><th>InvDate</th><th>LWkEnd</th></tr></thead><tbody><tr><td>28/03/2012</td><td>07/04/2012</td></tr><tr><td>10/12/2012</td><td>22/12/2012</td></tr><tr><td>5/2/2013</td><td>18/02/2013</td></tr><tr><td>31/3/2013</td><td>08/04/2013</td></tr><tr><td>19/5/2013</td><td>27/05/2013</td></tr><tr><td>15/9/2013</td><td>23/09/2013</td></tr><tr><td>11/12/2013</td><td>23/12/2013</td></tr><tr><td>2/3/2014</td><td>11/03/2014</td></tr><tr><td>14/5/2014</td><td>27/05/2014</td></tr><tr><td>13/6/2014</td><td>24/06/2014</td></tr><tr><td>7/7/2014</td><td>15/07/2014</td></tr><tr><td>4/8/2014</td><td>12/08/2014</td></tr></tbody></table>	InvDate	LWkEnd	28/03/2012	07/04/2012	10/12/2012	22/12/2012	5/2/2013	18/02/2013	31/3/2013	08/04/2013	19/5/2013	27/05/2013	15/9/2013	23/09/2013	11/12/2013	23/12/2013	2/3/2014	11/03/2014	14/5/2014	27/05/2014	13/6/2014	24/06/2014	7/7/2014	15/07/2014	4/8/2014	12/08/2014
InvDate	LWkEnd																										
28/03/2012	07/04/2012																										
10/12/2012	22/12/2012																										
5/2/2013	18/02/2013																										
31/3/2013	08/04/2013																										
19/5/2013	27/05/2013																										
15/9/2013	23/09/2013																										
11/12/2013	23/12/2013																										
2/3/2014	11/03/2014																										
14/5/2014	27/05/2014																										
13/6/2014	24/06/2014																										
7/7/2014	15/07/2014																										
4/8/2014	12/08/2014																										

### lunarweekname

Эта функция возвращает значение года и номер лунной недели, соответствующие метке времени первой миллисекунды первого дня лунной недели, содержащего значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

#### Синтаксис:

```
LunarWeekName (date [, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие лунные недели, положительные — последующие.
<b>first_ week_day</b>	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

**Примеры и результаты:**

Пример	Результат
<code>1unarweekname('12/01/2013')</code>	Возвращает 2006/02.
<code>1unarweekname('12/01/2013', -1)</code>	Возвращает 2006/01.
<code>1unarweekname('12/01/2013', 0, 1)</code>	Возвращает 2006/02.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере для каждой даты счета в таблице имя лунной недели создается из года, в котором находится эта неделя, и связанного с ней номера недели, смещенного на одну неделю путем указания для элемента <code>period_no</code> значения 1.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, LunarWeekName(InvDate, 1) AS LwkName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>lunarweekname()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>LWkName</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>2012/14</td></tr> <tr><td>10/12/2012</td><td>2012/51</td></tr> <tr><td>5/2/2013</td><td>2013/07</td></tr> <tr><td>31/3/2013</td><td>2013/14</td></tr> <tr><td>19/5/2013</td><td>2013/21</td></tr> <tr><td>15/9/2013</td><td>2013/38</td></tr> <tr><td>11/12/2013</td><td>2013/51</td></tr> <tr><td>2/3/2014</td><td>2014/10</td></tr> <tr><td>14/5/2014</td><td>2014/21</td></tr> <tr><td>13/6/2014</td><td>2014/25</td></tr> <tr><td>7/7/2014</td><td>2014/28</td></tr> <tr><td>4/8/2014</td><td>2014/32</td></tr> </tbody> </table>	InvDate	LWkName	28/03/2012	2012/14	10/12/2012	2012/51	5/2/2013	2013/07	31/3/2013	2013/14	19/5/2013	2013/21	15/9/2013	2013/38	11/12/2013	2013/51	2/3/2014	2014/10	14/5/2014	2014/21	13/6/2014	2014/25	7/7/2014	2014/28	4/8/2014	2014/32
InvDate	LWkName																										
28/03/2012	2012/14																										
10/12/2012	2012/51																										
5/2/2013	2013/07																										
31/3/2013	2013/14																										
19/5/2013	2013/21																										
15/9/2013	2013/38																										
11/12/2013	2013/51																										
2/3/2014	2014/10																										
14/5/2014	2014/21																										
13/6/2014	2014/25																										
7/7/2014	2014/28																										
4/8/2014	2014/32																										

### lunarweekstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

#### Синтаксис:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие лунные недели, положительные — последующие.
<b>first_week_day</b>	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

**Примеры и результаты:**

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>lunarweekstart('12/01/2013')</code>	Возвращает 08/01/2013.
<code>lunarweekstart('12/01/2013', -1)</code>	Возвращает 01/01/2013.
<code>lunarweekstart('12/01/2013', 0, 1 )</code>	Возвращает 09/01/2013. Поскольку смещение, определенное значением 1 для <b>first_week_day</b> , означает, что начало года изменяется на 02/01/2013.



Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня лунной недели каждой даты счета в таблице, где значение date смещается на одну неделю путем указания для элемента period_no значения 1.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, LunarWeekStart(InvDate, 1) AS LwkStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции lunarweekstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <tr> <th>InvDate</th><th>LWkStart</th></tr> <tr> <td>28/03/2012</td><td>01/04/2012</td></tr> <tr> <td>10/12/2012</td><td>16/12/2012</td></tr> <tr> <td>5/2/2013</td><td>12/02/2013</td></tr> <tr> <td>31/3/2013</td><td>02/04/2013</td></tr> <tr> <td>19/5/2013</td><td>21/05/2013</td></tr> <tr> <td>15/9/2013</td><td>17/09/2013</td></tr> <tr> <td>11/12/2013</td><td>17/12/2013</td></tr> <tr> <td>2/3/2014</td><td>05/03/2014</td></tr> <tr> <td>14/5/2014</td><td>21/05/2014</td></tr> <tr> <td>13/6/2014</td><td>18/06/2014</td></tr> <tr> <td>7/7/2014</td><td>09/07/2014</td></tr> <tr> <td>4/8/2014</td><td>06/08/2014</td></tr> </table>	InvDate	LWkStart	28/03/2012	01/04/2012	10/12/2012	16/12/2012	5/2/2013	12/02/2013	31/3/2013	02/04/2013	19/5/2013	21/05/2013	15/9/2013	17/09/2013	11/12/2013	17/12/2013	2/3/2014	05/03/2014	14/5/2014	21/05/2014	13/6/2014	18/06/2014	7/7/2014	09/07/2014	4/8/2014	06/08/2014
InvDate	LWkStart																										
28/03/2012	01/04/2012																										
10/12/2012	16/12/2012																										
5/2/2013	12/02/2013																										
31/3/2013	02/04/2013																										
19/5/2013	21/05/2013																										
15/9/2013	17/09/2013																										
11/12/2013	17/12/2013																										
2/3/2014	05/03/2014																										
14/5/2014	21/05/2014																										
13/6/2014	18/06/2014																										
7/7/2014	09/07/2014																										
4/8/2014	06/08/2014																										

### makedate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, месяц **MM** и день **DD**.

#### Синтаксис:

```
MakeDate (YYYY [ , MM [ , DD ] ] )
```

**Возвращаемые типы данных:** dual

#### 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).

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

Пример	Результат
<code>makedate(2012)</code>	возвращает 2012-01-01
<code>makedate(12)</code>	возвращает 0012-01-01
<code>makedate(2012,12)</code>	возвращает 2012-12-01
<code>makedate(2012,2,14)</code>	возвращает 2012-02-14

### maketime

Эта функция возвращает время, рассчитанное в формате часы **hh**, минуты **mm** и секунды **ss**.

**Синтаксис:**

```
MakeTime (hh [ , mm [ , ss ] ])
```

**Возвращаемые типы данных:** dual

**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.

Примеры и результаты:

Пример	Результат
<code>maketime( 22 )</code>	возвращает 22:00:00
<code>maketime( 22, 17 )</code>	возвращает 22:17:00
<code>maketime( 22, 17, 52 )</code>	возвращает 22:17:52

### makeweekdate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, неделя **WW** и день недели **D**.

**Синтаксис:**

```
MakeWeekDate (YYYY [ , WW [ , D ] ])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
YYYY	Год — целое число.
WW	Неделя — целое число.
D	День недели — целое число. Если день недели не задан, используется 0 (понедельник).

**Примеры и результаты:**

Пример	Результат
<code>makeweekdate(2014,6,6)</code>	возвращает 2014-02-09
<code>makeweekdate(2014,6,1)</code>	возвращает 2014-02-04
<code>makeweekdate(2014,6)</code>	возвращает 2014-02-03 (для недели допускается значение 0)

### minute

Эта функция возвращает время в минутах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

**Синтаксис:**

```
minute (expression)
```

**Возвращаемые типы данных:** целое число

**Примеры и результаты:**

Пример	Результат
<code>minute ( '09:14:36' )</code>	возвращает 14
<code>minute ( '0.5555' )</code>	возвращает 19 (так как 0,5555 = 13:19:55)

### month

Эта функция возвращает двойное значение с именем месяца, как определено переменной окружения **MonthNames**, и целое в диапазоне от 1 до 12. Месяц высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

**Синтаксис:**

```
month (expression)
```

**Возвращаемые типы данных:** dual

Примеры и результаты:

Пример	Результат
<code>month( '2012-10-12' )</code>	возвращает Oct (октябрь)
<code>month( '35648' )</code>	возвращает Aug (август), так как 35648 = 1997-08-06

### monthend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

**Синтаксис:**

```
MonthEnd(date[, period_no])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие месяцы, положительные — последующие.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>monthend('19/02/2012')</code>	Возвращает 29/02/2012 23:59:59.
<code>monthend('19/02/2001', -1)</code>	Возвращает 31/01/2001 23:59:59.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение последнего дня в месяце каждой даты счета в таблице, где базовая дата смещается на четыре месяца путем указания для элемента <i>period_no</i> значения 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthEnd(InvDate, 4) AS MthEnd Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>monthend()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>MthEnd</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>31/07/2012</td></tr> <tr><td>10/12/2012</td><td>30/04/2013</td></tr> <tr><td>5/2/2013</td><td>30/06/2013</td></tr> <tr><td>31/3/2013</td><td>31/07/2013</td></tr> <tr><td>19/5/2013</td><td>30/09/2013</td></tr> <tr><td>15/9/2013</td><td>31/01//2014</td></tr> <tr><td>11/12/2013</td><td>30/04//2014</td></tr> <tr><td>2/3/2014</td><td>31/07//2014</td></tr> <tr><td>14/5/2014</td><td>30/09/2014</td></tr> <tr><td>13/6/2014</td><td>31/10/2014</td></tr> <tr><td>7/7/2014</td><td>30/11/2014</td></tr> <tr><td>4/8/2014</td><td>31/12/2014</td></tr> </tbody> </table>	InvDate	MthEnd	28/03/2012	31/07/2012	10/12/2012	30/04/2013	5/2/2013	30/06/2013	31/3/2013	31/07/2013	19/5/2013	30/09/2013	15/9/2013	31/01//2014	11/12/2013	30/04//2014	2/3/2014	31/07//2014	14/5/2014	30/09/2014	13/6/2014	31/10/2014	7/7/2014	30/11/2014	4/8/2014	31/12/2014
InvDate	MthEnd																										
28/03/2012	31/07/2012																										
10/12/2012	30/04/2013																										
5/2/2013	30/06/2013																										
31/3/2013	31/07/2013																										
19/5/2013	30/09/2013																										
15/9/2013	31/01//2014																										
11/12/2013	30/04//2014																										
2/3/2014	31/07//2014																										
14/5/2014	30/09/2014																										
13/6/2014	31/10/2014																										
7/7/2014	30/11/2014																										
4/8/2014	31/12/2014																										

### monthname

Эта функция возвращает значение, отображающее месяц (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня указанного месяца.

#### Синтаксис:

```
MonthName (date[, period_no])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие месяцы, положительные — последующие.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>monthname('19/10/2013')</code>	Возвращает Oct 2013. Поскольку в этом и других примерах оператор <b>SET Monthnames</b> задан как Jan;Feb;Mar и т. д.
<code>monthname('19/10/2013', -1)</code>	Возвращает Sep 2013.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере для каждой даты счета в таблице имя месяца создается на основе имени месяца, смещенного на четыре месяца от поля <code>base_date</code> и на основе года.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthName(InvDate, 4) AS MthName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>monthname()</code>. В</p> <table><thead><tr><th>InvDate</th><th>MthName</th></tr></thead><tbody><tr><td>28/03/2012</td><td>Jul 2012</td></tr><tr><td>10/12/2012</td><td>Apr 2013</td></tr><tr><td>5/2/2013</td><td>Jun 2013</td></tr><tr><td>31/3/2013</td><td>Jul 2013</td></tr><tr><td>19/5/2013</td><td>Sep 2013</td></tr><tr><td>15/9/2013</td><td>Jan 2014</td></tr><tr><td>11/12/2013</td><td>Apr 2014</td></tr><tr><td>2/3/2014</td><td>Jul 2014</td></tr><tr><td>14/5/2014</td><td>Sep 2014</td></tr><tr><td>13/6/2014</td><td>Oct 2014</td></tr><tr><td>7/7/2014</td><td>Nov 2014</td></tr><tr><td>4/8/2014</td><td>Dec 2014</td></tr></tbody></table>	InvDate	MthName	28/03/2012	Jul 2012	10/12/2012	Apr 2013	5/2/2013	Jun 2013	31/3/2013	Jul 2013	19/5/2013	Sep 2013	15/9/2013	Jan 2014	11/12/2013	Apr 2014	2/3/2014	Jul 2014	14/5/2014	Sep 2014	13/6/2014	Oct 2014	7/7/2014	Nov 2014	4/8/2014	Dec 2014
InvDate	MthName																										
28/03/2012	Jul 2012																										
10/12/2012	Apr 2013																										
5/2/2013	Jun 2013																										
31/3/2013	Jul 2013																										
19/5/2013	Sep 2013																										
15/9/2013	Jan 2014																										
11/12/2013	Apr 2014																										
2/3/2014	Jul 2014																										
14/5/2014	Sep 2014																										
13/6/2014	Oct 2014																										
7/7/2014	Nov 2014																										
4/8/2014	Dec 2014																										

### monthsend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

#### Синтаксис:

```
MonthsEnd (n_months, date[, period_no [, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>n_months</b>	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
<b>date</b>	Дата для вычисления.
<b>period_no</b>	Период можно сместить, задав значение в поле <b>period_no</b> , целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие периоды, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
monthsend(4, '19/07/2013')	Возвращает 31/08/2013.
monthsend(4, '19/10/2013', -1)	Возвращает 31/08/2013.
monthsend(4, '19/10/2013', 0, 2)	Возвращает 31/01/2014. Поскольку началом года становится месяц 2.



Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение окончания последнего дня двухмесячного периода для каждой даты счета, смещенного вперед на один двухмесячный период.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthsEnd(2, InvDate, 1) AS BiMthsEnd Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции MonthsEnd().</p> <table> <thead> <tr> <th>InvDate</th><th>BiMthsEnd</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>30/06/2012</td></tr> <tr> <td>10/12/2012</td><td>28/02/2013</td></tr> <tr> <td>5/2/2013</td><td>30/04/2013</td></tr> <tr> <td>31/3/2013</td><td>30/04/2013</td></tr> <tr> <td>19/5/2013</td><td>31/08/2013</td></tr> <tr> <td>15/9/2013</td><td>31/12/2013</td></tr> <tr> <td>11/12/2013</td><td>28/02/2014</td></tr> <tr> <td>2/3/2014</td><td>30/06/2014</td></tr> <tr> <td>14/5/2014</td><td>31/08/2014</td></tr> <tr> <td>13/6/2014</td><td>31/08/2014</td></tr> <tr> <td>7/7/2014</td><td>31/10/2014</td></tr> <tr> <td>4/8/2014</td><td>31/10/2014</td></tr> </tbody> </table>	InvDate	BiMthsEnd	28/03/2012	30/06/2012	10/12/2012	28/02/2013	5/2/2013	30/04/2013	31/3/2013	30/04/2013	19/5/2013	31/08/2013	15/9/2013	31/12/2013	11/12/2013	28/02/2014	2/3/2014	30/06/2014	14/5/2014	31/08/2014	13/6/2014	31/08/2014	7/7/2014	31/10/2014	4/8/2014	31/10/2014
InvDate	BiMthsEnd																										
28/03/2012	30/06/2012																										
10/12/2012	28/02/2013																										
5/2/2013	30/04/2013																										
31/3/2013	30/04/2013																										
19/5/2013	31/08/2013																										
15/9/2013	31/12/2013																										
11/12/2013	28/02/2014																										
2/3/2014	30/06/2014																										
14/5/2014	31/08/2014																										
13/6/2014	31/08/2014																										
7/7/2014	31/10/2014																										
4/8/2014	31/10/2014																										

### monthsname

Эта функция возвращает значение, представляющее диапазон месяцев периода (форматированного согласно переменным скрипта **MonthNames**), а также года. Базовое числовое значение соответствует метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату.

#### Синтаксис:

```
MonthsName (n_months, date[, period_no[, first_month_of_year]])
```

## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>n_months</b>	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
<b>date</b>	Дата для вычисления.
<b>period_no</b>	Период можно сместить, задав значение в поле <b>period_no</b> , целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие периоды, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
monthsname(4, '19/10/2013')	Возвращает Sep-Dec 2013. Поскольку в этом и других примерах оператор <b>SET Monthnames</b> задан как Jan;Feb;Mar и т. д.
monthsname(4, '19/10/2013', -1)	Возвращает May-Aug 2013.
monthsname(4, '19/10/2013', 0, 2)	Возвращает Oct-Jan 2014. Поскольку год указан начинающимся в месяце 2, поэтому четырехмесячный период заканчивается в первом месяце следующего года.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере для каждой даты счета в таблице имя месяцев создается на основе диапазона месяцев в двухмесячном периоде и на основе года. Диапазон смещен месяцами 4x2 путем указания для элемента period_no значения 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthsName(2, InvDate, 4) AS MthsName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции monthsname().</p> <table> <thead> <tr> <th>InvDate</th><th>MthsName</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>Nov-Dec 2012</td></tr> <tr> <td>10/12/2012</td><td>Jul-Aug 2013</td></tr> <tr> <td>5/2/2013</td><td>Sep-Oct 2013</td></tr> <tr> <td>31/3/2013</td><td>Nov-Dec2013</td></tr> <tr> <td>19/5/2013</td><td>Jan-Feb 2014</td></tr> <tr> <td>15/9/2013</td><td>May-Jun 2014</td></tr> <tr> <td>11/12/2013</td><td>Jul-Aug 2014</td></tr> <tr> <td>2/3/2014</td><td>Nov-Dec 2014</td></tr> <tr> <td>14/5/2014</td><td>Jan-Feb 2015</td></tr> <tr> <td>13/6/2014</td><td>Jan-Feb 2015</td></tr> <tr> <td>7/7/2014</td><td>Mar-Apr 2015</td></tr> <tr> <td>4/8/2014</td><td>Mar-Apr 2015</td></tr> </tbody> </table>	InvDate	MthsName	28/03/2012	Nov-Dec 2012	10/12/2012	Jul-Aug 2013	5/2/2013	Sep-Oct 2013	31/3/2013	Nov-Dec2013	19/5/2013	Jan-Feb 2014	15/9/2013	May-Jun 2014	11/12/2013	Jul-Aug 2014	2/3/2014	Nov-Dec 2014	14/5/2014	Jan-Feb 2015	13/6/2014	Jan-Feb 2015	7/7/2014	Mar-Apr 2015	4/8/2014	Mar-Apr 2015
InvDate	MthsName																										
28/03/2012	Nov-Dec 2012																										
10/12/2012	Jul-Aug 2013																										
5/2/2013	Sep-Oct 2013																										
31/3/2013	Nov-Dec2013																										
19/5/2013	Jan-Feb 2014																										
15/9/2013	May-Jun 2014																										
11/12/2013	Jul-Aug 2014																										
2/3/2014	Nov-Dec 2014																										
14/5/2014	Jan-Feb 2015																										
13/6/2014	Jan-Feb 2015																										
7/7/2014	Mar-Apr 2015																										
4/8/2014	Mar-Apr 2015																										

### monthsstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

#### Синтаксис:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>n_months</b>	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
<b>date</b>	Дата для вычисления.
<b>period_no</b>	Период можно сместить, задав значение в поле <b>period_no</b> , целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле <b>base_date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие периоды, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
monthsstart(4, '19/10/2013')	Возвращает 1/09/2013.
monthsstart(4, '19/10/2013', -1)	Возвращает 01/05/2013.
monthsstart(4, '19/10/2013', 0, 2 )	Возвращает 01/10/2013. Поскольку началом года становится месяц 2.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня двухмесячного периода для каждой даты счета, смещенного вперед на один двухмесячный период.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthsStart(2, InvDate, 1) AS BiMthsStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции MonthsStart().</p> <table> <tr> <th>InvDate</th><th>BiMthsStart</th></tr> <tr> <td>28/03/2012</td><td>01/05/2012</td></tr> <tr> <td>10/12/2012</td><td>01/01/2013</td></tr> <tr> <td>5/2/2013</td><td>01/03/2013</td></tr> <tr> <td>31/3/2013</td><td>01/05/2013</td></tr> <tr> <td>19/5/2013</td><td>01/07/2013</td></tr> <tr> <td>15/9/2013</td><td>01/11/2013</td></tr> <tr> <td>11/12/2013</td><td>01/01/2014</td></tr> <tr> <td>2/3/2014</td><td>01/05/2014</td></tr> <tr> <td>14/5/2014</td><td>01/07/2014</td></tr> <tr> <td>13/6/2014</td><td>01/07/2014</td></tr> <tr> <td>7/7/2014</td><td>01/09/2014</td></tr> <tr> <td>4/8/2014</td><td>01/09/2014</td></tr> </table>	InvDate	BiMthsStart	28/03/2012	01/05/2012	10/12/2012	01/01/2013	5/2/2013	01/03/2013	31/3/2013	01/05/2013	19/5/2013	01/07/2013	15/9/2013	01/11/2013	11/12/2013	01/01/2014	2/3/2014	01/05/2014	14/5/2014	01/07/2014	13/6/2014	01/07/2014	7/7/2014	01/09/2014	4/8/2014	01/09/2014
InvDate	BiMthsStart																										
28/03/2012	01/05/2012																										
10/12/2012	01/01/2013																										
5/2/2013	01/03/2013																										
31/3/2013	01/05/2013																										
19/5/2013	01/07/2013																										
15/9/2013	01/11/2013																										
11/12/2013	01/01/2014																										
2/3/2014	01/05/2014																										
14/5/2014	01/07/2014																										
13/6/2014	01/07/2014																										
7/7/2014	01/09/2014																										
4/8/2014	01/09/2014																										

### monthstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

#### Синтаксис:

```
MonthStart(date[, period_no])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
<b>date</b>	Дата для вычисления.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>period_no</b>	<b>period_no</b> является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие месяцы, положительные — последующие.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат																										
<code>monthstart('19/10/2001')</code>	Возвращает 01/10/2001.																										
<code>monthstart('19/10/2001', -1)</code>	Возвращает 01/09/2001.																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня в месяце каждой даты счета в таблице, где <code>base_date</code> смещается на четыре месяца путем указания для элемента <code>period_no</code> значения 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, MonthStart(InvDate, 4) AS MthStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>monthstart()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>MthStart</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>01/07/2012</td></tr> <tr><td>10/12/2012</td><td>01/04/2013</td></tr> <tr><td>5/2/2013</td><td>01/06/2013</td></tr> <tr><td>31/3/2013</td><td>01/07/2013</td></tr> <tr><td>19/5/2013</td><td>01/09/2013</td></tr> <tr><td>15/9/2013</td><td>01/01/2014</td></tr> <tr><td>11/12/2013</td><td>01/04/2014</td></tr> <tr><td>2/3/2014</td><td>01/07/2014</td></tr> <tr><td>14/5/2014</td><td>01/09/2014</td></tr> <tr><td>13/6/2014</td><td>01/10/2014</td></tr> <tr><td>7/7/2014</td><td>01/11/2014</td></tr> <tr><td>4/8/2014</td><td>01/12/2014</td></tr> </tbody> </table>	InvDate	MthStart	28/03/2012	01/07/2012	10/12/2012	01/04/2013	5/2/2013	01/06/2013	31/3/2013	01/07/2013	19/5/2013	01/09/2013	15/9/2013	01/01/2014	11/12/2013	01/04/2014	2/3/2014	01/07/2014	14/5/2014	01/09/2014	13/6/2014	01/10/2014	7/7/2014	01/11/2014	4/8/2014	01/12/2014
InvDate	MthStart																										
28/03/2012	01/07/2012																										
10/12/2012	01/04/2013																										
5/2/2013	01/06/2013																										
31/3/2013	01/07/2013																										
19/5/2013	01/09/2013																										
15/9/2013	01/01/2014																										
11/12/2013	01/04/2014																										
2/3/2014	01/07/2014																										
14/5/2014	01/09/2014																										
13/6/2014	01/10/2014																										
7/7/2014	01/11/2014																										
4/8/2014	01/12/2014																										

### networkdays

Функция **networkdays** возвращает число рабочих дней (понедельник-пятница) между и включая значения, указанные в поле **start\_date** и **end\_date**, учитывая выходные, которые можно дополнительно задать в поле **holiday**.

#### Синтаксис:

```
networkdays (start_date, end_date [, holiday])
```

**Возвращаемые типы данных:** целое число

#### Аргументы:

Аргумент	Описание
<b>start_date</b>	Начальная дата для вычисления.
<b>end_date</b>	Конечная дата для вычисления.
<b>holiday</b>	Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми.  <b>Пример:</b> '25/12/2013', '26/12/2013'  Можно указать несколько периодов выходных дней, разделенных запятыми.  <b>Пример:</b> '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'

#### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>networkdays ('19/12/2013', '07/01/2014')</code>	Возвращает 14. В этом примере выходные дни не учитываются.
<code>networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013')</code>	Возвращает 12. В этом примере учитываются выходные в периоде с 25/12/2013 по 26/12/2013.
<code>networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014')</code>	Возвращает 10. В этом примере учитываются двухдневные периоды выходных дней.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																																																				
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>PayTable: LOAD recno() as InvID, * INLINE [ InvRec InvPaid 28/03/2012 28/04/2012 10/12/2012 01/01/2013 5/2/2013 5/3/2013 31/3/2013 01/5/2013 19/5/2013 12/6/2013 15/9/2013 6/10/2013 11/12/2013 12/01/2014 2/3/2014 2/4/2014 14/5/2014 14/6/2014 13/6/2014 14/7/2014 7/7/2014 14/8/2014 4/8/2014 4/9/2014 ] (delimiter is ' '); NrDays: Load *, NetworkDays(InvRec,InvPaid) As PaidDays Resident PayTable; Drop table PayTable;</pre>	<p>Результирующая таблица показывает возвращенные значения функции NetworkDays для каждой записи в таблице.</p> <table><tr><th>InvID</th><th>InvRec</th><th>InvPaid</th><th>PaidDays</th></tr><tr><td>1</td><td>28/03/2012</td><td>28/04/2012</td><td>23</td></tr><tr><td>2</td><td>10/12/2012</td><td>01/01/2013</td><td>17</td></tr><tr><td>3</td><td>5/2/2013</td><td>5/3/2013</td><td>21</td></tr><tr><td>4</td><td>31/3/2013</td><td>01/5/2013</td><td>23</td></tr><tr><td>5</td><td>19/5/2013</td><td>12/6/2013</td><td>18</td></tr><tr><td>6</td><td>15/9/2013</td><td>6/10/2013</td><td>15</td></tr><tr><td>7</td><td>11/12/2013</td><td>12/01/2014</td><td>23</td></tr><tr><td>8</td><td>2/3/2014</td><td>2/4/2014</td><td>23</td></tr><tr><td>9</td><td>14/5/2014</td><td>14/6/2014</td><td>23</td></tr><tr><td>10</td><td>13/6/2014</td><td>14/7/2014</td><td>22</td></tr><tr><td>11</td><td>7/7/2014</td><td>14/8/2014</td><td>29</td></tr><tr><td>12</td><td>4/8/2014</td><td>4/9/2014</td><td>24</td></tr></table>	InvID	InvRec	InvPaid	PaidDays	1	28/03/2012	28/04/2012	23	2	10/12/2012	01/01/2013	17	3	5/2/2013	5/3/2013	21	4	31/3/2013	01/5/2013	23	5	19/5/2013	12/6/2013	18	6	15/9/2013	6/10/2013	15	7	11/12/2013	12/01/2014	23	8	2/3/2014	2/4/2014	23	9	14/5/2014	14/6/2014	23	10	13/6/2014	14/7/2014	22	11	7/7/2014	14/8/2014	29	12	4/8/2014	4/9/2014	24
InvID	InvRec	InvPaid	PaidDays																																																		
1	28/03/2012	28/04/2012	23																																																		
2	10/12/2012	01/01/2013	17																																																		
3	5/2/2013	5/3/2013	21																																																		
4	31/3/2013	01/5/2013	23																																																		
5	19/5/2013	12/6/2013	18																																																		
6	15/9/2013	6/10/2013	15																																																		
7	11/12/2013	12/01/2014	23																																																		
8	2/3/2014	2/4/2014	23																																																		
9	14/5/2014	14/6/2014	23																																																		
10	13/6/2014	14/7/2014	22																																																		
11	7/7/2014	14/8/2014	29																																																		
12	4/8/2014	4/9/2014	24																																																		

### now

Эта функция возвращает метку текущего времени по системным часам. Значение по умолчанию — 1.


#### Синтаксис:

```
now([ timer_mode])
```



**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
timer_ mode	Может иметь следующие значения:  0 (время последней завершенной загрузки данных) 1 (время вызова функции) 2 (время открытия приложения)  <div> Если вы используете функцию в скрипте загрузки данных, функция <b>timer_mode=0</b> выдаст время последней завершенной загрузки данных, а <b>timer_mode=1</b> выдаст время вызова функции в текущей загрузке данных.</div>

**Примеры и результаты:**

Пример	Результат
now( 0)	Возвращает время завершения последней загрузки данных.
now( 1)	<ul style="list-style-type: none"><li>• При использовании в выражении визуализации будет возвращено время вызова функции.</li><li>• При использовании в скрипте загрузки данных будет возвращено время вызова функции в текущей загрузке данных.</li></ul>
now( 2)	Возвращает время открытия приложения.

### quarterend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

**Синтаксис:**

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает квартал, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие кварталы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>quarterend('29/10/2005')</code>	Возвращает 31/12/2005 23:59:59.
<code>quarterend('29/10/2005', -1)</code>	Возвращает 30/09/2005 23:59:59.
<code>quarterend('29/10/2005', 0, 3)</code>	Возвращает 30/11/2005 23:59:59.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение последнего дня в квартале каждой даты счета в таблице, где первый месяц в году указан как месяц 3.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, QuarterEnd(InvDate, 0, 3) AS QtrEnd Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>quarterend()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>QtrEnd</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>31/05/2012</td></tr> <tr><td>10/12/2012</td><td>28/02/2013</td></tr> <tr><td>5/2/2013</td><td>28/02/2013</td></tr> <tr><td>31/3/2013</td><td>31/05/2013</td></tr> <tr><td>19/5/2013</td><td>31/05/2013</td></tr> <tr><td>15/9/2013</td><td>30/11/2013</td></tr> <tr><td>11/12/2013</td><td>28/02/2014</td></tr> <tr><td>2/3/2014</td><td>31/05/2014</td></tr> <tr><td>14/5/2014</td><td>31/05/2014</td></tr> <tr><td>13/6/2014</td><td>31/08/2014</td></tr> <tr><td>7/7/2014</td><td>31/08/2014</td></tr> <tr><td>4/8/2014</td><td>31/08/2014</td></tr> </tbody> </table>	InvDate	QtrEnd	28/03/2012	31/05/2012	10/12/2012	28/02/2013	5/2/2013	28/02/2013	31/3/2013	31/05/2013	19/5/2013	31/05/2013	15/9/2013	30/11/2013	11/12/2013	28/02/2014	2/3/2014	31/05/2014	14/5/2014	31/05/2014	13/6/2014	31/08/2014	7/7/2014	31/08/2014	4/8/2014	31/08/2014
InvDate	QtrEnd																										
28/03/2012	31/05/2012																										
10/12/2012	28/02/2013																										
5/2/2013	28/02/2013																										
31/3/2013	31/05/2013																										
19/5/2013	31/05/2013																										
15/9/2013	30/11/2013																										
11/12/2013	28/02/2014																										
2/3/2014	31/05/2014																										
14/5/2014	31/05/2014																										
13/6/2014	31/08/2014																										
7/7/2014	31/08/2014																										
4/8/2014	31/08/2014																										

### quartername

Эта функция возвращает значение, отображающее месяцы квартала (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня квартала.

#### Синтаксис:

```
QuarterName (date[, period_no[, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает квартал, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие кварталы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

**Примеры и результаты:**

Пример	Результат
<code>quartername('29/10/2013')</code>	Возвращает Oct-Dec 2013.
<code>quartername('29/10/2013', -1)</code>	Возвращает Jul-Sep 2013.
<code>quartername('29/10/2013', 0, 3)</code>	Возвращает Sep-Nov 2013.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере для каждой даты счета в таблице имя квартала создается на основе квартала, содержащего <i>InvID</i>. Первый месяц в году указан как месяц 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, QuarterName(InvDate, 0, 4) AS QtrName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <i>quartername()</i>.</p> <table> <thead> <tr> <th>InvDate</th><th>QtrName</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>Jan-Mar 2011</td></tr> <tr> <td>10/12/2012</td><td>Oct-Dec 2012</td></tr> <tr> <td>5/2/2013</td><td>Jan-Mar 2012</td></tr> <tr> <td>31/3/2013</td><td>Jan-Mar 2012</td></tr> <tr> <td>19/5/2013</td><td>Apr-Jun 2013</td></tr> <tr> <td>15/9/2013</td><td>Jul-Sep 2013</td></tr> <tr> <td>11/12/2013</td><td>Oct-Dec 2013</td></tr> <tr> <td>2/3/2014</td><td>Jan-Mar 2013</td></tr> <tr> <td>14/5/2014</td><td>Apr-Jun 2014</td></tr> <tr> <td>13/6/2014</td><td>Apr-Jun 2014</td></tr> <tr> <td>7/7/2014</td><td>Jul-Sep 2014</td></tr> <tr> <td>4/8/2014</td><td>Jul-Sep 2014</td></tr> </tbody> </table>	InvDate	QtrName	28/03/2012	Jan-Mar 2011	10/12/2012	Oct-Dec 2012	5/2/2013	Jan-Mar 2012	31/3/2013	Jan-Mar 2012	19/5/2013	Apr-Jun 2013	15/9/2013	Jul-Sep 2013	11/12/2013	Oct-Dec 2013	2/3/2014	Jan-Mar 2013	14/5/2014	Apr-Jun 2014	13/6/2014	Apr-Jun 2014	7/7/2014	Jul-Sep 2014	4/8/2014	Jul-Sep 2014
InvDate	QtrName																										
28/03/2012	Jan-Mar 2011																										
10/12/2012	Oct-Dec 2012																										
5/2/2013	Jan-Mar 2012																										
31/3/2013	Jan-Mar 2012																										
19/5/2013	Apr-Jun 2013																										
15/9/2013	Jul-Sep 2013																										
11/12/2013	Oct-Dec 2013																										
2/3/2014	Jan-Mar 2013																										
14/5/2014	Apr-Jun 2014																										
13/6/2014	Apr-Jun 2014																										
7/7/2014	Jul-Sep 2014																										
4/8/2014	Jul-Sep 2014																										

### quarterstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

#### Синтаксис:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает квартал, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие кварталы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
quarterstart('29/10/2005')	Возвращает 01/10/2005.
quarterstart('29/10/2005', -1 )	Возвращает 01/07/2005.
quarterstart('29/10/2005', 0, 3)	Возвращает 01/09/2005.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня в квартале каждой даты счета в таблице, где первый месяц в году указан как месяц 3.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, QuarterStart(InvDate, 0, 3) AS QtrStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции quarterstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>QtrStart</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>01/03/2012</td></tr> <tr><td>10/12/2012</td><td>01/12/2012</td></tr> <tr><td>5/2/2013</td><td>01/12/2012</td></tr> <tr><td>31/3/2013</td><td>01/03/2013</td></tr> <tr><td>19/5/2013</td><td>01/03/2013</td></tr> <tr><td>15/9/2013</td><td>01/09/2013</td></tr> <tr><td>11/12/2013</td><td>01/12/2013</td></tr> <tr><td>2/3/2014</td><td>01/03/2014</td></tr> <tr><td>14/5/2014</td><td>01/03/2014</td></tr> <tr><td>13/6/2014</td><td>01/06/2014</td></tr> <tr><td>7/7/2014</td><td>01/06/2014</td></tr> <tr><td>4/8/2014</td><td>01/06/2014</td></tr> </tbody> </table>	InvDate	QtrStart	28/03/2012	01/03/2012	10/12/2012	01/12/2012	5/2/2013	01/12/2012	31/3/2013	01/03/2013	19/5/2013	01/03/2013	15/9/2013	01/09/2013	11/12/2013	01/12/2013	2/3/2014	01/03/2014	14/5/2014	01/03/2014	13/6/2014	01/06/2014	7/7/2014	01/06/2014	4/8/2014	01/06/2014
InvDate	QtrStart																										
28/03/2012	01/03/2012																										
10/12/2012	01/12/2012																										
5/2/2013	01/12/2012																										
31/3/2013	01/03/2013																										
19/5/2013	01/03/2013																										
15/9/2013	01/09/2013																										
11/12/2013	01/12/2013																										
2/3/2014	01/03/2014																										
14/5/2014	01/03/2014																										
13/6/2014	01/06/2014																										
7/7/2014	01/06/2014																										
4/8/2014	01/06/2014																										

### second

Эта функция возвращает время в секундах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

#### Синтаксис:

```
second (expression)
```

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
<code>second( '09:14:36' )</code>	возвращает 36
<code>second( '0.5555' )</code>	возвращает 55 (так как 0,5555 = 13:19:55)

### setdateyear

Данная функция берет в качестве входных значений **timestamp** и **year** и обновляет значение **timestamp** с учетом указанного входного значения **year**.

**Синтаксис:**

```
setdateyear (timestamp, year)
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>timestamp</b>	Стандартная метка времени Qlik Sense (часто просто дата).
<b>year</b>	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.

Пример	Результат
<code>setdateyear ( '29/10/2005', 2013)</code>	Возвращает '29/10/2013'
<code>setdateyear ( '29/10/2005 04:26:14', 2013)</code>	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	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre> 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 ]; </pre>	<p>Результирующая таблица содержит исходные даты и столбец, в котором для года было задано значение 2013.</p> <table> <tr> <th>testdates</th><th>NewYear</th></tr> <tr> <td>1/11/2012</td><td>1/11/2013</td></tr> <tr> <td>10/12/2012</td><td>10/12/2013</td></tr> <tr> <td>2/1/2012</td><td>2/1/2013</td></tr> <tr> <td>1/5/2013</td><td>1/5/2013</td></tr> <tr> <td>19/5/2013</td><td>19/5/2013</td></tr> <tr> <td>15/9/2013</td><td>15/9/2013</td></tr> <tr> <td>11/12/2013</td><td>11/12/2013</td></tr> <tr> <td>2/3/2014</td><td>2/3/2013</td></tr> <tr> <td>14/5/2014</td><td>14/5/2013</td></tr> <tr> <td>13/6/2014</td><td>13/6/2013</td></tr> <tr> <td>7/7/2014</td><td>7/7/2013</td></tr> <tr> <td>4/8/2014</td><td>4/8/2013</td></tr> </table>	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
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:** dual

#### Аргументы:

Аргумент	Описание
<b>timestamp</b>	Стандартная метка времени Qlik Sense (часто просто дата).
<b>year</b>	Четырехзначный год.
<b>month</b>	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**

## 5 Функции в скриптах и выражениях диаграммы

**DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат																						
<code>setdateyearmonth ('29/10/2005', 2013, 3)</code>	Возвращает '29/03/2013'																						
<code>setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)</code>	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».																						
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица содержит исходные даты и столбец, в котором для года было задано значение 2013.</p> <table><tr><th>testdates</th><th>NewYearMonth</th></tr><tr><td>1/11/2012</td><td>1/3/2013</td></tr><tr><td>10/12/2012</td><td>10/3/2013</td></tr><tr><td>2/1/2012</td><td>2/3/2013</td></tr><tr><td>19/5/2013</td><td>19/3/2013</td></tr><tr><td>15/9/2013</td><td>15/3/2013</td></tr><tr><td>11/12/2013</td><td>11/3/2013</td></tr><tr><td>14/5/2014</td><td>14/3/2013</td></tr><tr><td>13/6/2014</td><td>13/3/2013</td></tr><tr><td>7/7/2014</td><td>7/3/2013</td></tr><tr><td>4/8/2014</td><td>4/3/2013</td></tr></table>	testdates	NewYearMonth	1/11/2012	1/3/2013	10/12/2012	10/3/2013	2/1/2012	2/3/2013	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
testdates	NewYearMonth																						
1/11/2012	1/3/2013																						
10/12/2012	10/3/2013																						
2/1/2012	2/3/2013																						
19/5/2013	19/3/2013																						
15/9/2013	15/3/2013																						
11/12/2013	11/3/2013																						
14/5/2014	14/3/2013																						
13/6/2014	13/3/2013																						
7/7/2014	7/3/2013																						
4/8/2014	4/3/2013																						

### timezone

Эта функция возвращает имя текущего часового пояса, соответствующее имени, используемому в Windows.

#### Синтаксис:

```
TimeZone ( )
```

**Возвращаемые типы данных:** строка

#### Пример:

```
timezone( )
```

### today


Эта функция возвращает текущую дату по системным часам.

#### Синтаксис:

```
today ([ timer_mode])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
timer_ mode	<p>Может иметь следующие значения:</p> <ul style="list-style-type: none"><li>0 (день последней завершенной загрузки данных)</li><li>1 (день вызова функции)</li><li>2 (день открытия приложения)</li></ul> <div> Если вы используете функцию в скрипте загрузки данных, функция <b>timer_mode=0</b> выдаст день последней завершенной загрузки данных, а <b>timer_mode=1</b> выдаст день текущей загрузки данных.</div>

#### Примеры и результаты:

Пример	Результат
today( 0)	Возвращает день последней завершенной загрузки данных.
today( 1)	При использовании в выражении визуализации будет возвращен день вызова функции.  При использовании в скрипте загрузки данных будет возвращен день начала текущей загрузки данных.
today( 2)	Возвращает день открытия приложения.

### UTC

Возвращает текущее время Coordinated Universal Time.

#### Синтаксис:

```
UTC ( )
```

**Возвращаемые типы данных:** dual

**Пример:**

```
utc( )
```

### week

Эта функция возвращает номер недели в виде целого числа согласно стандарту ISO 8601. Номер недели высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

**Синтаксис:**

```
week (timestamp [, first_week_day [, broken_weeks [, reference_day]])
```

**Возвращаемые типы данных:** целое число

Аргумент	Описание
<b>timestamp</b>	Дата для вычисления в виде метки времени или выражения, определяемого по метке времени, для преобразования, например '2012-10-12'.
<b>first_week_day</b>	<p>Если не указать <b>first_week_day</b>, значение переменной <b>FirstWeekDay</b> будет использовано как первый день недели.</p> <p>Если необходимо использовать другой день в качестве первого дня недели, установите для элемента <b>first_week_day</b> следующее значение:</p> <ul style="list-style-type: none"><li>• 0 для понедельника</li><li>• 1 для вторника</li><li>• 2 для среды</li><li>• 3 для четверга</li><li>• 4 для пятницы</li><li>• 5 для субботы</li><li>• 6 для воскресенья</li></ul> <p>Целое число, возвращенное этой функцией, теперь будет использовать первый день недели, заданный параметром <b>first_week_day</b>.</p>

Аргумент	Описание
<b>broken_weeks</b>	<p>Если параметр <b>broken_weeks</b> не указан, значение переменной <b>BrokenWeeks</b> будет использовано для определения, какими должны быть недели: целыми или разбитыми.</p> <p>По умолчанию в функциях Qlik Sense используются целые недели. Это означает следующее:</p> <ul style="list-style-type: none"> <li>В одних годах 1-я неделя начинается в декабре, а в других годах 52-я или 53-я неделя заканчивается в январе.</li> <li>В 1-ой неделе всегда не менее четырех дней в январе.</li> </ul> <p>В качестве альтернативы можно использовать разбиение недель.</p> <ul style="list-style-type: none"> <li>52-я или 53-я неделя не будет продолжена в январе следующего года.</li> <li>1-я неделя будет начинаться 1 января и в большинстве случаев она будет неполной.</li> </ul> <p>Могут использоваться следующие значения:</p> <ul style="list-style-type: none"> <li>0 (= использовать целые недели)</li> <li>1 (= использовать разбитые недели)</li> </ul>
<b>reference_day</b>	<p>Если параметр <b>reference_day</b> не указан, значение переменной <b>ReferenceDay</b> 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.</p> <p>Используйте следующие значения, чтобы задать день ссылки:</p> <ul style="list-style-type: none"> <li>1 (= январь 1)</li> <li>2 (= январь 2)</li> <li>3 (= январь 3)</li> <li>4 (= январь 4)</li> <li>5 (= январь 5)</li> <li>6 (= январь 6)</li> <li>7 (= январь 7)</li> </ul>

Примеры и результаты:

Пример	Результат
<code>week( '2012-10-12' )</code>	возвращает 41.
<code>week( '35648' )</code>	возвращает 32, так как 35 648 = 1997–08–06
<code>week('2012-10-12', 0, 1)</code>	возвращает 42

### weekday

Эта функция возвращает двойное значение со следующим:

- Имя дня, как определено переменной окружения **DayNames**.
- Целое от 0 до 6, соответствующее номинальному дню недели (0–6).

**Синтаксис:**

```
weekday (date [,first_week_day=0])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>first_week_day</b>	<p>Если не указать <b>first_week_day</b>, значение переменной <b>FirstWeekDay</b> будет использовано как первый день недели.</p> <p>Если необходимо использовать другой день в качестве первого дня недели, установите для элемента <b>first_week_day</b> следующее значение:</p> <ul style="list-style-type: none"><li>• 0 для понедельника</li><li>• 1 для вторника</li><li>• 2 для среды</li><li>• 3 для четверга</li><li>• 4 для пятницы</li><li>• 5 для субботы</li><li>• 6 для воскресенья</li></ul> <p>Целое число, возвращенное этой функцией, теперь будет использовать первый день недели, заданный в элементе <b>first_week_day</b> как основной (0).</p> <p>См.: <i>FirstWeekDay</i> (страница 140)</p>

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

Если не указано иначе, в этих примерах для элемента **FirstWeekDay** установлено значение 0.

Пример	Результат
<code>weekday( '1971-10-12' )</code>	возвращает 'Tue' (вторник) и 1
<code>weekday( '1971-10-12' , 6)</code>	возвращает 'Tue' (вторник) и 2.  В этом примере мы используем воскресенье (6) в качестве первого дня недели.
<code>SET FirstWeekDay = 6;</code> ... <code>weekday( '1971-10-12' )</code>	возвращает 'Tue' (вторник) и 2.

### weekend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последней даты (воскресенья) календарной недели, включающей дату, заданную в поле **date**. По умолчанию для вывода используется формат даты **DateFormat**, установленный в скрипте.

**Синтаксис:**

```
WeekEnd (date [, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>shift</b> — целое число, где 0 обозначает неделю, включающую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>shift</b> , означают предшествующие недели, положительные — последующие.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<b>first_week_day</b>	Указывает день начала недели. Если не указано, используется значение переменной <b>FirstWeekDay</b> .  Возможные значения <b>first_week_day</b> : <ul style="list-style-type: none"><li>• 0 для понедельника</li><li>• 1 для вторника</li><li>• 2 для среды</li><li>• 3 для четверга</li><li>• 4 для пятницы</li><li>• 5 для субботы</li><li>• 6 для воскресенья</li></ul> См.: <i>FirstWeekDay</i> (страница 140)

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>weekend('10/01/2013')</code>	Возвращает 12/01/2013 23:59:59.
<code>weekend('10/01/2013', -1)</code>	Возвращает 06/01/2013 23:59:59.
<code>weekend('10/01/2013', 0, 1)</code>	Возвращает 14/01/2013 23:59:59.



Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение последнего дня недели после недели с датами каждого счета в таблице.</p> <pre>TempTable: LOAD ResNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, WeekEnd(InvDate, 1) AS WkEnd Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции weekend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>WkEnd</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>08/04/2012</td></tr> <tr> <td>10/12/2012</td><td>23/12/2012</td></tr> <tr> <td>5/2/2013</td><td>17/02/2013</td></tr> <tr> <td>31/3/2013</td><td>07/04/2013</td></tr> <tr> <td>19/5/2013</td><td>26/05/2013</td></tr> <tr> <td>15/9/2013</td><td>22/09/2013</td></tr> <tr> <td>11/12/2013</td><td>22/12/2013</td></tr> <tr> <td>2/3/2014</td><td>09/03/2014</td></tr> <tr> <td>14/5/2014</td><td>25/05/2014</td></tr> <tr> <td>13/6/2014</td><td>22/06/2014</td></tr> <tr> <td>7/7/2014</td><td>20/07/2014</td></tr> <tr> <td>4/8/2014</td><td>17/08/2014</td></tr> </tbody> </table>	InvDate	WkEnd	28/03/2012	08/04/2012	10/12/2012	23/12/2012	5/2/2013	17/02/2013	31/3/2013	07/04/2013	19/5/2013	26/05/2013	15/9/2013	22/09/2013	11/12/2013	22/12/2013	2/3/2014	09/03/2014	14/5/2014	25/05/2014	13/6/2014	22/06/2014	7/7/2014	20/07/2014	4/8/2014	17/08/2014
InvDate	WkEnd																										
28/03/2012	08/04/2012																										
10/12/2012	23/12/2012																										
5/2/2013	17/02/2013																										
31/3/2013	07/04/2013																										
19/5/2013	26/05/2013																										
15/9/2013	22/09/2013																										
11/12/2013	22/12/2013																										
2/3/2014	09/03/2014																										
14/5/2014	25/05/2014																										
13/6/2014	22/06/2014																										
7/7/2014	20/07/2014																										
4/8/2014	17/08/2014																										

### weekname

Эта функция возвращает значение года и номер недели с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня недели, содержащего значение, указанное в поле **date**.

#### Синтаксис:

```
WeekName (date[, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>shift</b> — целое число, где 0 обозначает неделю, включающую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>shift</b> , означают предшествующие недели, положительные — последующие.
<b>first_week_day</b>	Указывает день начала недели. Если не указано, используется значение переменной <b>FirstWeekDay</b> .  Возможные значения <b>first_week_day</b> : <ul style="list-style-type: none"><li>• 0 для понедельника</li><li>• 1 для вторника</li><li>• 2 для среды</li><li>• 3 для четверга</li><li>• 4 для пятницы</li><li>• 5 для субботы</li><li>• 6 для воскресенья</li></ul> См.: <i>FirstWeekDay</i> (страница 140)

**Примеры и результаты:**

Пример	Результат
<code>weekname('12/01/2013')</code>	Возвращает 2013/02.
<code>weekname('12/01/2013', -1)</code>	Возвращает 2013/01.
<code>weekname('12/01/2013', 0, 1)</code>	Возвращает 2013/02.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере для каждой даты счета в таблице имя недели создается на основе года, в котором находится эта неделя, и связанного с ней номера недели, смещенного на одну неделю путем указания для элемента <code>period_no</code> значения 1.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, weekName(InvDate, 1) AS wkName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции <code>weekname()</code>. Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>WkName</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>2012/14</td></tr> <tr><td>10/12/2012</td><td>2012/51</td></tr> <tr><td>5/2/2013</td><td>2013/07</td></tr> <tr><td>31/3/2013</td><td>2013/14</td></tr> <tr><td>19/5/2013</td><td>2013/21</td></tr> <tr><td>15/9/2013</td><td>2013/38</td></tr> <tr><td>11/12/2013</td><td>2013/51</td></tr> <tr><td>2/3/2014</td><td>2014/10</td></tr> <tr><td>14/5/2014</td><td>2014/21</td></tr> <tr><td>13/6/2014</td><td>2014/25</td></tr> <tr><td>7/7/2014</td><td>2014/29</td></tr> <tr><td>4/8/2014</td><td>2014/33</td></tr> </tbody> </table>	InvDate	WkName	28/03/2012	2012/14	10/12/2012	2012/51	5/2/2013	2013/07	31/3/2013	2013/14	19/5/2013	2013/21	15/9/2013	2013/38	11/12/2013	2013/51	2/3/2014	2014/10	14/5/2014	2014/21	13/6/2014	2014/25	7/7/2014	2014/29	4/8/2014	2014/33
InvDate	WkName																										
28/03/2012	2012/14																										
10/12/2012	2012/51																										
5/2/2013	2013/07																										
31/3/2013	2013/14																										
19/5/2013	2013/21																										
15/9/2013	2013/38																										
11/12/2013	2013/51																										
2/3/2014	2014/10																										
14/5/2014	2014/21																										
13/6/2014	2014/25																										
7/7/2014	2014/29																										
4/8/2014	2014/33																										

### weekstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня (понедельника) календарной недели, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

#### Синтаксис:

```
WeekStart(date [, period_no[, first_week_day]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>shift</b> — целое число, где 0 обозначает неделю, включающую значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>shift</b> , означают предшествующие недели, положительные — последующие.
<b>first_week_day</b>	Указывает день начала недели. Если не указано, используется значение переменной <b>FirstWeekDay</b> .  Возможные значения <b>first_week_day</b> : <ul style="list-style-type: none"><li>• 0 для понедельника</li><li>• 1 для вторника</li><li>• 2 для среды</li><li>• 3 для четверга</li><li>• 4 для пятницы</li><li>• 5 для субботы</li><li>• 6 для воскресенья</li></ul> См.: <i>FirstWeekDay</i> (страница 140)

**Примеры и результаты:**

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
<code>weekstart('12/01/2013')</code>	Возвращает 07/01/2013.
<code>weekstart('12/01/2013', -1 )</code>	Возвращает 31/11/2012.
<code>weekstart('12/01/2013', 0, 1)</code>	Возвращает 08/01/2013.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня недели после недели с датами каждого счета в таблице.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, weekStart(InvDate, 1) AS wkStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции weekstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table><thead><tr><th>InvDate</th><th>WkStart</th></tr></thead><tbody><tr><td>28/03/2012</td><td>02/04/2012</td></tr><tr><td>10/12/2012</td><td>17/12/2012</td></tr><tr><td>5/2/2013</td><td>11/02/2013</td></tr><tr><td>31/3/2013</td><td>01/04/2013</td></tr><tr><td>19/5/2013</td><td>20/05/2013</td></tr><tr><td>15/9/2013</td><td>16/09/2013</td></tr><tr><td>11/12/2013</td><td>16/12/2013</td></tr><tr><td>2/3/2014</td><td>03/03/2014</td></tr><tr><td>14/5/2014</td><td>19/05/2014</td></tr><tr><td>13/6/2014</td><td>16/06/2014</td></tr><tr><td>7/7/2014</td><td>14/07/2014</td></tr><tr><td>4/8/2014</td><td>11/08/2014</td></tr></tbody></table>	InvDate	WkStart	28/03/2012	02/04/2012	10/12/2012	17/12/2012	5/2/2013	11/02/2013	31/3/2013	01/04/2013	19/5/2013	20/05/2013	15/9/2013	16/09/2013	11/12/2013	16/12/2013	2/3/2014	03/03/2014	14/5/2014	19/05/2014	13/6/2014	16/06/2014	7/7/2014	14/07/2014	4/8/2014	11/08/2014
InvDate	WkStart																										
28/03/2012	02/04/2012																										
10/12/2012	17/12/2012																										
5/2/2013	11/02/2013																										
31/3/2013	01/04/2013																										
19/5/2013	20/05/2013																										
15/9/2013	16/09/2013																										
11/12/2013	16/12/2013																										
2/3/2014	03/03/2014																										
14/5/2014	19/05/2014																										
13/6/2014	16/06/2014																										
7/7/2014	14/07/2014																										
4/8/2014	11/08/2014																										

### weekyear

Эта функция возвращает год, которому принадлежит номер недели согласно стандарту ISO 8601. Номер недели в году может быть установлен в пределах от 1 до 52.

#### Синтаксис:

```
weekyear (expression)
```

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
<code>weekyear( '1996-12-30' )</code>	возвращает 1997, поскольку неделя 1 1998 года начинается 1996–12–30
<code>weekyear( '1997-01-02' )</code>	возвращает 1997
<code>weekyear( '1997-12-28' )</code>	возвращает 1997
<code>weekyear( '1997-12-30' )</code>	возвращает 1998, поскольку неделя 1 1998 года начинается 1997–12–29
<code>weekyear( '1999-01-02' )</code>	возвращает 1998, поскольку неделя 53 1998 года оканчивается 1999–01–03

**Ограничения:**

В определенные годы первая неделя начинается в декабре, например, в декабре 1997 года. В другие годы первая неделя начинается с 53-й недели предыдущего года, как, например, в январе 1999 года. В течение этих нескольких дней, когда номер недели текущего года относится к предыдущему году, функции **year** и **weekyear** возвращают разные значения.

### year

Эта функция возвращает год в виде целого числа, а выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

**Синтаксис:**

```
year (expression)
```

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
<code>year( '2012-10-12' )</code>	возвращает 2012
<code>year( '35648' )</code>	возвращает 1997, так как 35648 = 1997–08–06

### yearend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня года, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

### Синтаксис:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Возвращаемые типы данных: dual

### Аргументы:

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает год, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие годы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
yearend ( '19/10/2001' )	Возвращает 31/12/2001 23:59:59.
yearend ( '19/10/2001', -1 )	Возвращает 31/12/2000 23:59:59.
yearend ( '19/10/2001', 0, 4)	Возвращает 31/03/2002 23:59:59.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение последнего дня в году каждой даты счета в таблице, где первый месяц в году указан как месяц 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, YearEnd(InvDate, 0, 4) AS YrEnd Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>YrEnd</th></tr> </thead> <tbody> <tr> <td>28/03/2012</td><td>31/03/2011</td></tr> <tr> <td>10/12/2012</td><td>31/03/2012</td></tr> <tr> <td>5/2/2013</td><td>31/03/2013</td></tr> <tr> <td>31/3/2013</td><td>31/03/2013</td></tr> <tr> <td>19/5/2013</td><td>31/03/2014</td></tr> <tr> <td>15/9/2013</td><td>31/03/2014</td></tr> <tr> <td>11/12/2013</td><td>31/03/2014</td></tr> <tr> <td>2/3/2014</td><td>31/03/2014</td></tr> <tr> <td>14/5/2014</td><td>31/03/2015</td></tr> <tr> <td>13/6/2014</td><td>31/03/2015</td></tr> <tr> <td>7/7/2014</td><td>31/03/2015</td></tr> <tr> <td>4/8/2014</td><td>31/03/2015</td></tr> </tbody> </table>	InvDate	YrEnd	28/03/2012	31/03/2011	10/12/2012	31/03/2012	5/2/2013	31/03/2013	31/3/2013	31/03/2013	19/5/2013	31/03/2014	15/9/2013	31/03/2014	11/12/2013	31/03/2014	2/3/2014	31/03/2014	14/5/2014	31/03/2015	13/6/2014	31/03/2015	7/7/2014	31/03/2015	4/8/2014	31/03/2015
InvDate	YrEnd																										
28/03/2012	31/03/2011																										
10/12/2012	31/03/2012																										
5/2/2013	31/03/2013																										
31/3/2013	31/03/2013																										
19/5/2013	31/03/2014																										
15/9/2013	31/03/2014																										
11/12/2013	31/03/2014																										
2/3/2014	31/03/2014																										
14/5/2014	31/03/2015																										
13/6/2014	31/03/2015																										
7/7/2014	31/03/2015																										
4/8/2014	31/03/2015																										

### yearname

Эта функция возвращает 4-значное значение года с базовым числовым значением, соответствующим метке времени с первой миллисекундой первого дня года, содержащего значение, указанное в поле **date**.

#### Синтаксис:

```
YearName (date[, period_no[, first_month_of_year]] )
```



**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает год, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие годы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> . Отображаемое значение будет строчным, показывающим два года.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
yearname ( '19/10/2001' )	Возвращает 2001.
yearname ( '19/10/2001', -1 )	Возвращает 2000.
yearname ( '19/10/2001', 0, 4 )	Возвращает 2001-2002.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере создается цифровое имя четыре-плюс-четыре для годов, в которых обнаруживается каждая дата счета в таблице. Это является следствием того, что первый месяц в году указан как месяц 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, YearName(InvDate, 0, 4) AS YrName Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearname().</p> <table><thead><tr><th>InvDate</th><th>YrName</th></tr></thead><tbody><tr><td>28/03/2012</td><td>2011-2012</td></tr><tr><td>10/12/2012</td><td>2012-2013</td></tr><tr><td>5/2/2013</td><td>2012-2013</td></tr><tr><td>31/3/2013</td><td>2012-2013</td></tr><tr><td>19/5/2013</td><td>2013-2014</td></tr><tr><td>15/9/2013</td><td>2013-2014</td></tr><tr><td>11/12/2013</td><td>2013-2014</td></tr><tr><td>2/3/2014</td><td>2013-2014</td></tr><tr><td>14/5/2014</td><td>2014-2015</td></tr><tr><td>13/6/2014</td><td>2014-2015</td></tr><tr><td>7/7/2014</td><td>2014-2015</td></tr><tr><td>4/8/2014</td><td>2014-2015</td></tr></tbody></table>	InvDate	YrName	28/03/2012	2011-2012	10/12/2012	2012-2013	5/2/2013	2012-2013	31/3/2013	2012-2013	19/5/2013	2013-2014	15/9/2013	2013-2014	11/12/2013	2013-2014	2/3/2014	2013-2014	14/5/2014	2014-2015	13/6/2014	2014-2015	7/7/2014	2014-2015	4/8/2014	2014-2015
InvDate	YrName																										
28/03/2012	2011-2012																										
10/12/2012	2012-2013																										
5/2/2013	2012-2013																										
31/3/2013	2012-2013																										
19/5/2013	2013-2014																										
15/9/2013	2013-2014																										
11/12/2013	2013-2014																										
2/3/2014	2013-2014																										
14/5/2014	2014-2015																										
13/6/2014	2014-2015																										
7/7/2014	2014-2015																										
4/8/2014	2014-2015																										

### yearstart

Эта функция возвращает метку времени, соответствующую началу первого дня года, содержащего значение **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

### Синтаксис:

```
YearStart(date[, period_no[, first_month_of_year]])
```

**Возвращаемые типы данных:** dual

### Аргументы:

Аргумент	Описание
<b>date</b>	Дата для вычисления.
<b>period_no</b>	<b>period_no</b> — целое число, где 0 обозначает год, включающий значение, указанное в поле <b>date</b> . Отрицательные значения, заданные в поле <b>period_no</b> , означают предшествующие годы, положительные — последующие.
<b>first_month_of_year</b>	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле <b>first_month_of_year</b> .

### Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
yearstart ('19/10/2001')	Возвращает 01/01/2001.
yearstart ('19/10/2001', -1)	Возвращает 01/01/2000.
yearstart ('19/10/2001', 0, 4)	Возвращает 01/04/2001.

Пример	Результат																										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>В этом примере выполняется обнаружение первого дня в году каждой даты счета в таблице, где первый месяц в году указан как месяц 4.</p> <pre>TempTable: LOAD RecNo() as InvID, * Inline [ InvDate 28/03/2012 10/12/2012 5/2/2013 31/3/2013 19/5/2013 15/9/2013 11/12/2013 2/3/2014 14/5/2014 13/6/2014 7/7/2014 4/8/2014 ];  InvoiceData: LOAD *, YearStart(InvDate, 0, 4) AS YrStart Resident TempTable; Drop table TempTable;</pre>	<p>Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.</p> <table> <thead> <tr> <th>InvDate</th><th>YrStart</th></tr> </thead> <tbody> <tr><td>28/03/2012</td><td>01/04/2011</td></tr> <tr><td>10/12/2012</td><td>01/04/2012</td></tr> <tr><td>5/2/2013</td><td>01/04/2012</td></tr> <tr><td>31/3/2013</td><td>01/04/2012</td></tr> <tr><td>19/5/2013</td><td>01/04/2013</td></tr> <tr><td>15/9/2013</td><td>01/04/2013</td></tr> <tr><td>11/12/2013</td><td>01/04/2013</td></tr> <tr><td>2/3/2014</td><td>01/04/2013</td></tr> <tr><td>14/5/2014</td><td>01/04/2014</td></tr> <tr><td>13/6/2014</td><td>01/04/2014</td></tr> <tr><td>7/7/2014</td><td>01/04/2014</td></tr> <tr><td>4/8/2014</td><td>01/04/2014</td></tr> </tbody> </table>	InvDate	YrStart	28/03/2012	01/04/2011	10/12/2012	01/04/2012	5/2/2013	01/04/2012	31/3/2013	01/04/2012	19/5/2013	01/04/2013	15/9/2013	01/04/2013	11/12/2013	01/04/2013	2/3/2014	01/04/2013	14/5/2014	01/04/2014	13/6/2014	01/04/2014	7/7/2014	01/04/2014	4/8/2014	01/04/2014
InvDate	YrStart																										
28/03/2012	01/04/2011																										
10/12/2012	01/04/2012																										
5/2/2013	01/04/2012																										
31/3/2013	01/04/2012																										
19/5/2013	01/04/2013																										
15/9/2013	01/04/2013																										
11/12/2013	01/04/2013																										
2/3/2014	01/04/2013																										
14/5/2014	01/04/2014																										
13/6/2014	01/04/2014																										
7/7/2014	01/04/2014																										
4/8/2014	01/04/2014																										

### yeartodate

Эта функция определяет, находится ли введенная метка времени в том году, в котором находится дата последней загрузки скрипта, и возвращает значение True, если это так, и False если это не так.

#### Синтаксис:

```
YearToDate (timestamp [ , yearoffset [ , firstmonth [ , todaydate] ] ] )
```

#### Возвращаемые типы данных: Boolean

Если дополнительные параметры не используются, то значение данной функции может быть любой датой в пределах одного календарного года с 1 января до даты последнего выполнения скрипта включительно.

### Аргументы:

Аргумент	Описание
timestamp	Метка времени, подлежащая оценке, например '2012-10-12'.
yearoffset	При указании элемента <b>yearoffset</b> , элемент <b>yeartodate</b> возвращает значение True для того же периода в другом году. Отрицательное значение смещения <b>yearoffset</b> указывает предыдущий год, положительное значение смещения — будущий год. Наиболее поздняя дата с начала года до последнего момента достигается путем указания <b>yearoffset</b> = -1. Если значение не указано, принимается 0.
firstmonth	Если в поле <b>firstmonth</b> 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 <b>firstmonth</b> = 5.
todaydate	Задав значение <b>todaydate</b> (метка времени последнего выполнения скрипта, если не указано), можно сместить день, используемый в качестве верхней границы периода.

### Примеры и результаты:

В следующих примерах предполагается время последней перезагрузки = 2011-11-18

Пример	Результат
yeartodate( '2010-11-18')	возвращает False
yeartodate( '2011-02-01')	возвращает True
yeartodate( '2011-11-18')	возвращает True
yeartodate( '2011-11-19')	возвращает False
yeartodate( '2011-11-19', 0, 1, '2011-12-31')	возвращает True
yeartodate( '2010-11-18', -1)	возвращает True
yeartodate( '2011-11-18', -1)	возвращает False
yeartodate( '2011-04-30', 0, 5)	возвращает False
yeartodate( '2011-05-01', 0, 5)	возвращает True

## 5.6 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:**

Элемент `log10(3)` возвращает 0,4771

### **pow**

Возвращает  $x$  в степени  $y$ . Результат — число.

```
pow ( x, y )
```

#### **Примеры и результаты:**

Элемент `pow(3, 3)` возвращает 27

### **sqr**

Возвращает  $x$  в квадрате ( $x$  в степени 2). Результат — число.

```
sqr ( x )
```

#### **Примеры и результаты:**

Элемент `sqr(3)` возвращает 9

### **sqrt**

Квадратный корень из  $x$ . Функция определена, только если  $x \geq 0$ . Результат — положительное

число.

```
sqrt(x )
```

### Примеры и результаты:

Элемент sqrt(3) возвращает 1,732

## 5.7 Функции поля

Эти функции могут использоваться только в выражениях диаграмм.

Функции полей возвращают целые числа или строки, выявляя различные аспекты выборок поля.

### Функции счетчика

GetSelectedCount

**GetSelectedCount()** находит выбранные (зеленые) значения в поле.

```
GetSelectedCount — функция диаграммы (field_name [, include_excluded])
```

GetAlternativeCount

**GetAlternativeCount()** используется для обнаружения альтернативных (светло-серых) значений в указанном поле.

```
GetAlternativeCount — функция диаграммы (field_name)
```

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)
```

GetExcludedCount

**GetExcludedCount()** находит исключенные (темно-серые) значения в указанном поле.

```
GetExcludedCount — функция диаграммы (страница 498) (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])
```

### Функции поля и выборки

GetCurrentSelections

**GetCurrentSelections()** возвращает текущие выборки в приложении.

```
GetCurrentSelections — функция диаграммы ([record_sep [,tag_sep [,value_sep  
[,max_values]]]])
```

GetFieldSelections

Функция **GetFieldSelections()** возвращает **string** для текущих выборок в поле.

```
GetFieldSelections — функция диаграммы ( field_name [, value_sep [, max_  
values]])
```

### GetAlternativeCount — функция диаграммы

**GetAlternativeCount()** используется для обнаружения альтернативных (светло-серых) значений в указанном поле.

**Синтаксис:**

```
GetAlternativeCount (field_name)
```

**Возвращаемые типы данных:** целое число

**Аргументы:**

Аргумент	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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  GetAlternativeCount ([First name])	Значение 4, поскольку существует 4 уникальных и исключенных (серых) значения в элементе <b>First name</b> .
При условии выбора элементов <b>John</b> и <b>Peter</b> .  GetAlternativeCount ([First name])	Значение 3, поскольку существует 3 уникальных и исключенных (серых) значения в элементе <b>First name</b> .



Примеры	Результаты
<p>При условии, что в элементе <b>First name</b> значения не выбраны.</p> <p>GetAlternativeCount ([First name])</p>	<p>Значение 0, поскольку выборки нет.</p>

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 текущие выборки в приложении.

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	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.

## 5 Функции в скриптах и выражениях диаграммы

Аргументы	Описание
state_name	Имя другого состояния, выбранное для определенной визуализации. В приложении Qlik Engine API настроены другие состояния. Если используется аргумент <b>state_name</b> , 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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> . <code>GetCurrentSelections ()</code>	'First name: John'
При условии выбора элементов <b>John</b> и <b>Peter</b> в элементе <b>First name</b> . <code>GetCurrentSelections ()</code>	'First name: John, Peter'
При условии выбора элементов <b>John</b> и <b>Peter</b> в элементе <b>First name</b> и выбора элемента <b>JA</b> в элементе <b>Initials</b> . <code>GetCurrentSelections ()</code>	'First name: John, Peter  Initials: JA'
При условии выбора элемента <b>John</b> в элементе <b>First name</b> , а <b>JA</b> в элементе <b>Initials</b> . <code>GetCurrentSelections ( chr(13)&amp;chr(10) , ' = ' )</code>	'First name = John  Initials = JA'
При условии выбора всех имен, кроме Sue, в элементе <b>First name</b> и отсутствии выборов в элементе <b>Initials</b> . <code>GetCurrentSelections (chr(13)&amp;chr(10), '=', ', ' ,3)</code>	'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 исключенные (темно-серые) значения в указанном поле.

### Синтаксис:

**GetExcludedCount** (field\_name)

**Возвращаемые типы данных:** строка

**Ограничения:**

Функция **GetExcludedCount()** оценивает только поля со связанными значениями, то есть поля без выборок. Для полей с выборками **GetExcludedCount()** будет возвращено значение 0.

**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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  GetExcludedCount ([Initials])	Значение 5, поскольку существует 5 исключенных (серых) значений в элементе <b>Initials</b> . Шестая ячейка (JA) будет белой, поскольку она связана с выборкой John в элементе <b>First name</b> .
При условии выбора элементов <b>John</b> и <b>Peter</b> .  GetExcludedCount ([Initials])	Значение 3, поскольку элемент Peter связан с 2 значениями в элементе <b>Initials</b> .
При условии, что в элементе <b>First name</b> значения не выбраны.  GetExcludedCount ([Initials])	Значение 0, поскольку выборок нет.
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  GetExcludedCount ([First name])	Значение 0, поскольку функция <b>GetExcludedCount()</b> оценивает только поля со связанными значениями, то есть поля без выборок.

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 **string** для текущих выборок 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

#### Аргументы:

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.
value_sep	Разделитель значений в поле. По умолчанию используется «, ».
max_values	Максимальное число отдельно отображаемых значений, введенных в поле. При вводе большого числа значений используется формат «x из y значений». По умолчанию установлено 6.
state_name	Имя другого состояния, выбранное для определенной визуализации. В приложении Qlik Engine API настроены другие состояния. Если используется аргумент <b>state_name</b> , 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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  <code>GetFieldSelections ([First name])</code>	«John»
При условии выбора элементов <b>John</b> и <b>Peter</b> .  <code>GetFieldSelections ([First name])</code>	«John,Peter»
При условии выбора элементов <b>John</b> и <b>Peter</b> .  <code>GetFieldSelections ([First name],'; ')</code>	«John; Peter»
При условии выбора элементов <b>John, Sue, Mark</b> в элементе <b>First name</b> .  <code>GetFieldSelections ([First name],';',2)</code>	«NOT Jane;Peter», поскольку значение 2 указано, как значение аргумента <code>max_values</code> . 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:

Аргумент	Описание
fieldname	Имя поля для оценки.

Аргумент	Описание
includeexcluded	Если для элемента <b>includeexcluded</b> установлено значение True, число будет включать в себя выбранные значения, исключенные выборками в другом поле.

### Примеры:

```
GetNotSelectedCount( Country )  
GetNotSelectedCount( Country, true )
```

## 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 число

### Синтаксис:

```
GetPossibleCount (field_name)
```

### 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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  <code>GetPossibleCount ([Initials])</code>	Значение 1, поскольку в элементе «Инициалы» значение 1 связано с выборкой, элементом <b>John</b> , в элементе <b>First name</b> .
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  <code>GetPossibleCount ([First name])</code>	Значение 1, поскольку существует 1 выборка, элемент <b>John</b> , в элементе <b>First name</b> .

Примеры	Результаты
<p>При условии, что элемент <b>Peter</b> выбран в элементе <b>First name</b>.</p> <p><code>GetPossibleCount ([Initials])</code></p>	<p>Значение 2, поскольку элемент Peter связан с 2 значениями в элементе <b>Initials</b>.</p>
<p>При условии, что в элементе <b>First name</b> значения не выбраны.</p> <p><code>GetPossibleCount ([First name])</code></p>	<p>Значение 5, поскольку выборки нет, но есть 5 уникальных значений в элементе <b>First name</b>.</p>
<p>При условии, что в элементе <b>First name</b> значения не выбраны.</p> <p><code>GetPossibleCount ([Initials])</code></p>	<p>Значение 6, поскольку выборки нет, но есть 6 уникальных значений в элементе <b>Initials</b>.</p>

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 число

**Аргументы:**

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.
include_excluded	Если установлено значение <b>True()</b> , 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 другого состояния, выбранное для определенной визуализации. В приложении Qlik Engine API настроены другие состояния. Если используется аргумент <b>state_name</b> , 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	Результаты
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  <code>GetSelectedCount ([First name])</code>	Значение 1, поскольку одно значение выбрано в элементе <b>First name</b> .
При условии, что элемент <b>John</b> выбран в элементе <b>First name</b> .  <code>GetSelectedCount ([Initials])</code>	Значение 0, поскольку в элементе <b>Initials</b> значения не выбраны.
При отсутствии выборов в элементе <b>First name</b> выберите все значения в элементе <b>Initials</b> , а затем выберите значение <b>Yes</b> в элементе <b>Has cellphone</b> .  <code>GetSelectedCount ([Initials])</code>	6. Хотя при выборках элемента <b>Initials</b> MC и PD имеют значение <b>Has cellphone</b> , заданное как <b>No</b> , результатом все равно будет 6, поскольку для аргумента <code>include_excluded</code> задано значение <code>True()</code> .

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 '|');
```

## 5.8 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.



---

## 5 Функции в скриптах и выражениях диаграммы

---

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)
```

### ConnectionString

Функция **ConnectionString()** возвращает имя активного подключения данных для подключений ODBC или OLE DB. Функция возвращает пустую строку, если не выполнен оператор **connect**, или после оператора **disconnect**.

```
ConnectionString ()
```

### FileNameBase

Функция **FileNameBase** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути или расширения.

```
FileNameBase ()
```

### FileDir

Функция **FileDir** возвращает строку, содержащую путь к каталогу табличного файла, читаемого в текущий момент.

```
FileDir ()
```

### FileExtension

Функция **FileExtension** возвращает строку, содержащую расширение табличного файла, читаемого в текущий момент.

```
FileExtension ()
```

### FileName

Функция **FileName** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути, но с расширением.

```
FileName ()
```

### FilePath

Функция **FilePath** возвращает строку, содержащую полный путь табличного файла, читаемого в текущий момент.

```
FilePath ()
```

### FileSize

Функция **FileSize** возвращает целое, содержащее размер в байтах файла filename, или, если не указан файл filename, табличного файла, читаемого в текущий момент.

```
FileSize ()
```

### FileTime

Функция **FileTime** возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.

```
FileTime ([ filename ])
```

### GetFolderPath

Функция **GetFolderPath** возвращает значение функции Microsoft Windows *SHGetFolderPath*. Данная функция берет в качестве значения ввода имя папки Microsoft Windows и возвращает полный путь папки.

```
GetFolderPath ()
```

### QvdCreateTime

Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.

```
QvdCreateTime (filename)
```

### QvdFieldName

Эта функция скрипта возвращает имя числа поля **fieldno**, если оно существует в файле QVD, в противном случае — 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:

Аргумент	Описание
filename	<p>Имя медиафайла, включающее при необходимости путь, в качестве подключения к данным папки.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p>
attributename	Имя мета-тега.

В примерах используется функция **GetFolderPath** для обнаружения путей к медиафайлам. Поскольку **GetFolderPath** поддерживается только в устаревшем режиме, необходимо заменить ссылки на **GetFolderPath** путем для подключения к данным lib://.

См.: *Ограничение доступа к файловой системе (страница 698)*

### Пример 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
```

## 5 Функции в скриптах и выражениях диаграммы

---

```
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, // Unsynchronized 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
```

### Пример 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,
```

## 5 Функции в скриптах и выражениях диаграммы

---

```
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,
```

## 5 Функции в скриптах и выражениях диаграммы

---

```
// 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,
```

## 5 Функции в скриптах и выражениях диаграммы

---

```
// 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;
```



---

## 5 Функции в скриптах и выражениях диаграммы

---

```
// examples: 0=No correction, 1=Differential correction,
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

### Пример 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
```

### Пример 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 данных для подключений ODBC или OLE DB. Функция возвращает пустую строку, если не выполнен оператор **connect**, или после оператора **disconnect**.

#### Синтаксис:

```
ConnectString()
```

Примеры и результаты:

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:

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:

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:

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:

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.*

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

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:**

Аргумент	Описание
filename	<p>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.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

Примеры и результаты:

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://myData/xyz.xls' )	Возвращает размер файла xyz.xls.

### FileTime

Функция **FileTime** возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.

#### Синтаксис:

```
FileTime ([ filename ])
```

#### Аргументы:

Аргумент	Описание
filename	<p>Имя файла, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

#### Примеры и результаты:

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:

Аргумент	Описание
<b>foldername</b>	Имя папки Microsoft Windows.  Имя папки не должно содержать пробелов. Из имени папки, отображающегося в Windows Explorer, следует удалить все пробелы.  Примеры:  <i>MyMusic</i>  <i>MyDocuments</i>

### Примеры и результаты:

Назначение этого примера — получить пути следующих папок Microsoft Windows: *MyMusic*, *MyPictures* и *Windows*. Добавьте образец скрипта в свое приложение и перезагрузите.

```
LOAD  
  GetFolderPath('MyMusic') as MyMusic,  
  GetFolderPath('MyPictures') as MyPictures,  
  GetFolderPath('Windows') as Windows  
AutoGenerate 1;
```

После перезагрузки приложения в модель данных будут добавлены поля *MyMusic*, *MyPictures* и *Windows*. Каждое поле содержит путь к папке, определенной во время ввода. Пример.

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

## QvdCreateTime

Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.

### Синтаксис:

**QvdCreateTime (filename)**

### Аргументы:

Аргумент	Описание
filename	<p>Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

### Пример:

```
QvdCreateTime('MyFile.qvd')
QvdCreateTime('C:\MyDir\MyFile.qvd')
QvdCreateTime('lib://data\MyFile.qvd')
```

## QvdFieldName

Эта функция скрипта возвращает имя числа поля **fieldno**, если оно существует в файле QVD, в противном случае — значение NULL.

### Синтаксис:

```
QvdFieldName (filename , fieldno)
```

### Аргументы:

Аргумент	Описание
filename	<p>Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>
fieldno	Номер поля (с отсчетом от 0) внутри таблицы, находящейся в файле QVD.

### Примеры:

```
QvdFieldName ('MyFile.qvd', 3)
QvdFieldName ('C:\MyDir\MyFile.qvd', 5)
QvdFieldName ('lib://data\MyFile.qvd', 5)
```

## QvdNoOfFields

Эта функция скрипта возвращает число полей в файле QVD.

### Синтаксис:

```
QvdNoOfFields (filename)
```



### Аргументы:

Аргумент	Описание
filename	<p>Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

### Примеры:

```
QvdNoOfFields ('MyFile.qvd')
QvdNoOfFields ('C:\MyDir\MyFile.qvd')
QvdNoOfFields ('lib://data\MyFile.qvd')
```

## QvdNoOfRecords

Эта функция скрипта возвращает число записей, находящихся в настоящее время в файле QVD.

### Синтаксис:

```
QvdNoOfRecords (filename)
```

### Аргументы:

Аргумент	Описание
filename	<p>Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

### Примеры:

```
QvdNoOfRecords ('myFile.qvd')  
QvdNoOfRecords ('C:\myDir\myFile.qvd')  
QvdNoOfRecords ('lib://data\myFile.qvd')
```

## QvdTableName

Эта функция скрипта возвращает имя таблицы, хранящейся в файле QVD.

### Синтаксис:

```
QvdTableName (filename)
```

### Аргументы:

Аргумент	Описание
filename	<p>Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла.</p> <p><b>Пример: 'lib://Table Files/'</b></p> <p>В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются:</p> <ul style="list-style-type: none"><li>• абсолютный</li></ul> <p><b>Пример: c:\data\</b></p> <ul style="list-style-type: none"><li>• относительно рабочего каталога приложения Qlik Sense.</li></ul> <p><b>Пример: data\</b></p> <ul style="list-style-type: none"><li>• URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети.</li></ul> <p><b>Пример: http://www.qlik.com</b></p>

### Примеры:

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

## 5.9 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.

```
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)
```

## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** число

**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:**

Пример	Результат
<code>BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')</code>  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 ] ])
```

## 5 Функции в скриптах и выражениях диаграммы

**Возвращаемые типы данных:** числовой. Результат имеет числовой денежный формат по умолчанию. .

**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 <b>pv</b> 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 <b>type</b> отсутствует, он соответствует 0.

Примеры и результаты:

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:** число

**Аргументы:**

Аргумент	Описание
rate	Процентная ставка за период.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
<code>nper</code>	Итоговое число сроков оплаты аннуитета.
<code>pmt</code>	Оплата, совершаемая в каждый период. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.
<code>pv</code>	Текущая стоимость или единовременно выплачиваемая сумма, которая соответствует ряду будущих выплат в настоящее время. Если <b>pv</b> отсутствует, он соответствует 0 (нулю).
<code>fv</code>	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если <b>fv</b> отсутствует, он соответствует 0.
<code>type</code>	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если <b>type</b> отсутствует, он соответствует 0.

Примеры и результаты:

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?  <code>nPer(0.005, -20, 0, 800)</code>	Возвращает 36,56

### Pmt

Эта функция возвращает платеж по кредиту на основе периодических, постоянных платежей и постоянной процентной ставки.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

**Возвращаемые типы данных:** числовой. Результат имеет числовой денежный формат по умолчанию. .

Чтобы узнать итоговую сумму, уплаченную в течение действия кредита, умножьте возвращенное значение **pmt** на **nper**.

**Аргументы:**

Аргумент	Описание
<code>rate</code>	Процентная ставка за период.
<code>nper</code>	Итоговое число сроков оплаты аннуитета.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
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 <b>pv</b> отсутствует, он соответствует 0 (нулю).
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если <b>fv</b> отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если <b>type</b> отсутствует, он соответствует 0.

Примеры и результаты:

Пример	Результат
Следующая формула возвращает ежемесячный платеж по кредиту в 20 тыс. долл. США под 10% годовых, которые необходимо выплатить за 8 месяцев:  <code>Pmt(0.1/12, 8, 20000)</code>	Возвращает -\$2,594.66
Для того же кредита, если платеж необходимо осуществлять в начале периода, размер платежа составляет:  <code>Pmt(0.1/12, 8, 20000, 0, 1)</code>	Возвращает -\$2,573.21

### PV

Эта функция возвращает текущую стоимость вложения.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

**Возвращаемые типы данных:** числовой. Результат имеет числовой денежный формат по умолчанию. .

Текущая стоимость — это итоговая сумма, которая соответствует ряду будущих выплат в настоящее время. Например, если Вы берете деньги в займы, для кредитора сумма кредита является текущей стоимостью.

**Аргументы:**

Аргумент	Описание
rate	Процентная ставка за период.
nper	Итоговое число сроков оплаты аннуитета.



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
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 <b>fv</b> отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если <b>type</b> отсутствует, он соответствует 0.

Примеры и результаты:

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%?  <code>PV(0.07/12, 12*5, -100, 0, 0)</code>	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:**

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 <b>pv</b> отсутствует, он соответствует 0 (нулю).
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если <b>fv</b> отсутствует, он соответствует 0.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если <b>type</b> отсутствует, он соответствует 0.

Примеры и результаты:

Пример	Результат
Какова процентная ставка 5-летнего кредита по аннуитету в размере 10 тыс. долл. США с ежемесячными платежами 300 долл. США?  <code>Rate(60, -300, 10000)</code>	Возвращает 2.00%

### 5.10 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.



*Для большей ясности 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

## 5 Функции в скриптах и выражениях диаграммы

---

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()** преобразует формат выражения в цифровой формат, установленный в системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования), дополнительно разделяет десятые и сотые доли.

```
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:

р [Функции интерпретации \(страница 562\)](#)

### ApplyCodepage

**ApplyCodepage()** применяет другой набор символов кодовой страницы к полю или тексту, указанному в выражении. Аргумент **codepage** должен иметь числовой формат.



*Несмотря на то что функцию `ApplyCodepage` можно использовать в выражениях диаграммы, наиболее часто она используется в качестве 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:

Аргумент	Описание
text	Поле или текст, к которому необходимо применить другую кодовую страницу, заданную аргументом <b>codepage</b> .
codepage	Число, представляющее кодовую страницу для применения к полю или выражению, заданному параметром <b>text</b> .

#### Примеры и результаты:

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 разные наборы символов: кириллица, иврит и др., из формата UTF-8. В этой ситуации может потребоваться загрузка по строкам с применением разных кодовых страниц к каждой строке.  Значение <b>codepage</b> 1253 представляет греческий набор символов Windows, значение 1255 — иврит, а значение 65001 — латинские символы UTF-8.

См. также: *Набор символов (страница 99)*

### 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:** dual

#### 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	Результаты	Параметр 1	Параметр 2
Date( A ) , где A=35648	Строка:	97-08-06	8/6/97
	Число:	35648	35648
Date( A, 'YY.MM.DD' ) , где A=35648	Строка:	97.08.06	97.08.06
	Число:	35648	35648
Date( A, 'DD.MM.YYYY' ) , где A=35648.375	Строка:	06.08.1997	06.08.1997
	Число:	35648.375	35648.375
Date( A, 'YY.MM.DD' ) , где A=8/6/97	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:** dual

#### 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:

Пример	Описание
<p>Добавьте следующие примеры в скрипт и запустите его.</p> <pre>Load dual (   NameDay, NumDay ) as   DayOfWeek inline   [ NameDay, NumDay     Monday, 0     Tuesday, 1     Wednesday, 2     Thursday, 3     Friday, 4     Saturday, 5     Sunday, 6 ];</pre>	<p>Поле DayOfWeek можно использовать в визуализации, например в качестве измерения. В таблице дни недели автоматически сортируются в правильной числовой последовательности, а не по алфавиту.</p>
<pre>Load Dual('Q' &amp;   Ceil(Month(Now())/3), Ceil(Month(Now())/3)) as   Quarter   AutoGenerate 1;</pre>	<p>В этом примере выполняется определение текущего квартала. Это значение отображается как Q1, если функция <b>Now()</b> 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.</p>
<pre>Dual('Q' &amp; Ceil(   Month(Date)/3),   Ceil(Month(   Date)/3)) as   Quarter</pre>	<p>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.</p>
<pre>Dual(WeekYear(Date)   &amp; '-w' &amp; week   (Date), weekStart   (Date)) as YearWeek</pre>	<p>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.</p>

### 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:** dual

**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: .

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:** dual

**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 MoneyFormat1: kr ##0,00, MoneyThousandSep'
- Параметр MoneyFormat 2: \$ #,##0.00, MoneyThousandSep','

Пример	Результаты	Параметр 1	Параметр 2
Money( A ) где A=35 648	Строка:	kr 35 648,00	35 648,00 \$
	Число:	35 648,00	35 648,00
Money( A, '#,##0 ¥', '.' , ',' ) где A=3 564 800	Строка:	3 564 800 ¥	3 564 800 ¥
	Число:	3 564 800	3 564 800

## Num

**Num()** преобразует формат выражения в цифровой 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:**

```
Num (number[, format[, dec_sep [, thou_sep]])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая, как будет отформатирована полученная строка даты. Если игнорируется, то используется формат даты, используемый 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.

**Примеры и результаты:**

В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр формата числа 1: `###0`
- Параметр формата числа 2: `#,###0`

Пример	Результаты	Параметр 1	Параметр 2
Num( A, '0.0' ) где A=35 648,375	Строка:	35 648 375	35 648,375
	Число:	35 648 375	35 648,375
Num( A, '#,##0.##', '.', ',' ) где A=35 648	Строка:	35 648,00	35 648,00
	Число:	35 648	35 648
Num( pi( ), '0,00' )	Строка:	3,14	003
	Число:	3,141592653	3,141592653

### Пример

Добавьте образец скрипта в свое приложение и запустите.

Затем создайте прямую таблицу, используя field1 и field2 в качестве измерений.

```
Sheet1:
let result= Num( pi( ), '0,00' );
Load * inline
[field1; field2
9; 8,2
1; $(result)
](delimiter is ';');
```

### Результат

Элемент field1 содержит значения 1 и 9.

Элемент field2 содержит значения 3,14 и 8,2.

### 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:** dual

#### 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	Результаты	Параметр 1	Параметр 2
Time( A ) где A=0,375	Строка:	09:00:00	09.00.00
	Число:	0,375	0,375
Time( A ) где A=35 648,375	Строка:	09:00:00	09.00.00
	Число:	35 648,375	35 648,375
Time( A, 'hh-mm' ) где A=0,99999	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:** dual

**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	Результаты	Параметр 1	Параметр 2
Timestamp( A ) где A=35 648,375	Строка:	97-08-06 09:00:00	8/6/97 09:00:00
	Число:	35 648,375	35 648,375
Timestamp( A, 'YYYY-MM-DD hh.mm' ) где A=35 648	Строка:	1997-08-06 00.00	1997-08-06 00.00
	Число:	35 648	35 648

### 5.11 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()** используется для определения количества 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.

---

## 5 Функции в скриптах и выражениях диаграммы

---

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:  $\text{combin}(p,q) = p! / q!(p-q)!$ , 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:  $\text{permut}(p,q) = (p)! / (p - q)!$ , 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** .

### BitCount

**BitCount()** используется для определения количества 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 число

### Примеры и результаты:

Примеры	Результаты
BitCount ( 3 )	3 является двоичным числом 101, 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:** число

### Аргументы:

Аргумент	Описание
<b>x</b>	Число ввода.
<b>step</b>	Приращение интервала. Значение по умолчанию — 1.
<b>offset</b>	Определяет базовое значение интервала шага. Значение по умолчанию — 0.

### 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, <b>2 &lt; x &lt;=3</b> , 3 < x <=4...
Ceil(4.2 )	Возвращает 5

Примеры	Результаты
<code>ceil(3.88 ,0.1)</code>	Возвращает 3,9  В данном примере значение размера интервала — 0,1, базовое значение интервала — 0.  Интервалы: ... 3.7 < x <= 3.8, <b>3.8 &lt; x &lt;= 3.9</b> , 3.9 < x <= 4.0...
<code>ceil(3.88 ,5)</code>	Возвращает 5
<code>ceil(1.1 ,1)</code>	Возвращает 2
<code>ceil(1.1 ,1,0.5)</code>	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: ... <b>0.5 &lt; x &lt;=1.5</b> , 1.5 < x <= 2.5, 2.5 < x <=3.5, 3.5 < x <=4.5...
<code>ceil(1.1 ,1,-0.01)</code>	Возвращает 1,99  Интервалы: ... -0.01 < x <= 0.99, <b>0.99 &lt; x &lt;= 1.99</b> , 1.99 < x <=2.99...

### Combin

**Combin()** возвращает число комбинаций **q** элементов, которые могут быть получены из набора элементов **p**. Как видно из формулы:  $\text{combin}(p,q) = p! / q!(p-q)!$ , порядок выбора элементов не имеет значения.

#### Синтаксис:

**Combin** (p, q)

**Возвращаемые типы данных:** целое число

#### Ограничения:

Нецелые элементы будут усечены.

#### Примеры и результаты:

Примеры	Результаты
Сколько сочетаний 7 чисел может быть получено из 35 чисел лотереи?  <code>combin( 35,7 )</code>	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 число

#### 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:** Boolean

#### 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:** число

### Примеры и результаты:

Примеры	Результаты
<code>fabs( 2.4 )</code>	Возвращает 2,4
<code>fabs( -3.8 )</code>	Возвращает 3,8

## Fact

**Fact()** возвращает факториал положительного целого числа **x**.

### Синтаксис:

```
Fact (x)
```

**Возвращаемые типы данных:** целое число

### Ограничения:

Если число **x** не является целым, оно будет обрезано. Неположительные числа возвращают значение NULL.

### Примеры и результаты:

Примеры	Результаты
<code>Fact( 1 )</code>	Возвращает 1
<code>Fact( 5 )</code>	Возвращает 120 (1 * 2 * 3 * 4 * 5 = 120)
<code>Fact( -5 )</code>	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:** число

**Аргументы:**

Аргумент	Описание
<b>x</b>	Число ввода.
<b>step</b>	Приращение интервала. Значение по умолчанию — 1.
<b>offset</b>	Определяет базовое значение интервала шага. Значение по умолчанию — 0.

**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, <b>2 &lt;= x &lt; 3</b> , 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, <b>3.8 &lt;= x &lt; 3.9</b> , 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: ... <b>0.5 &lt;= x &lt; 1.5</b> , 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:** число

**Аргументы:**

Аргумент	Описание
<b>a</b>	Делимое
<b>b</b>	Делитель

**Примеры и результаты:**

Примеры	Результаты
<code>fmod( 7,2 )</code>	Возвращает 1
<code>fmod( 7.5,2 )</code>	Возвращает 1,5
<code>fmod( 9,3 )</code>	Возвращает 0
<code>fmod( -4,3 )</code>	Возвращает -1
<code>fmod( 4,-3 )</code>	Возвращает 1
<code>fmod( -4,-3 )</code>	Возвращает -1

## Frac

**Frac()** возвращает дробную часть **x**.

Десятичная дробь определяется следующим образом:  $\text{Frac}(x) + \text{Floor}(x) = x$ . Говоря простым языком, это значит, что дробная часть положительного числа является разницей между числом (x) и целым числом, предшествующим ему.

Например: дробная часть  $11,43 = 11,43 - 11 = 0,43$

Для отрицательного числа допустим, что  $-1,4$ ,  $\text{Floor}(-1.4) = -2$ , что приведет к следующему результату.

Дробная часть  $-1,4 = 1,4 - (-2) = -1,4 + 2 = 0,6$

**Синтаксис:**

```
Frac (x)
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
<b>x</b>	Число, для которого возвращается дробная часть.

### Примеры и результаты:

Примеры	Результаты
<code>Frac( 11.43 )</code>	Возвращает 0,43
<code>Frac( -1.4 )</code>	Возвращает 0,6

## 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 число

### Ограничения:

Значение **integer\_number2** должно быть больше 0.

### Примеры и результаты:

Примеры	Результаты
<code>Mod( 7,2 )</code>	Возвращает 1
<code>Mod( 7.5,2 )</code>	Возвращает NULL
<code>Mod( 9,3 )</code>	Возвращает 0
<code>Mod( -4,3 )</code>	Возвращает 2
<code>Mod( 4,-3 )</code>	Возвращает NULL
<code>Mod( -4,-3 )</code>	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:** Boolean

**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:  $\text{permut}(p, q) = (p)! / (p - q)!$ , порядок выбора элементов имеет значение.

**Синтаксис:**

```
Permut (p, q)
```

**Возвращаемые типы данных:** целое число

**Ограничения:**

Нецелые аргументы будут усечены.

**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: число



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:

Аргумент	Описание
<b>x</b>	Число ввода.
<b>step</b>	Приращение интервала. Значение по умолчанию — 1.
<b>offset</b>	Определяет базовое значение интервала шага. Значение по умолчанию — 0.

### 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, <b>3 &lt;= x &lt;4</b> ...
Round(3.8,4)	Возвращает 4
Round(2.5)	Возвращает 3. Округляется в большую сторону, поскольку значение 2,5 находится ровно посередине интервала шага по умолчанию.
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: ... <b>0 &lt;= x &lt;4</b> , 4 <= x <8, 8 <= x <12

Примеры	Результаты
Round(2,6)	<p>Возвращает 0. Округляется в меньшую сторону, поскольку значение 2 меньше половины интервала шага, равного 6.</p> <p>В данном примере значение размера шага — 6, базовое значение интервала шага — 0.</p> <p>Интервалы: ...<b>0</b> &lt;= x &lt;<b>6</b>, 6 &lt;= x &lt;12, 12&lt;= x &lt;18</p>
Round(3.88,0.1)	<p>Возвращает 3,9</p> <p>В данном примере значение размера шага — 0,1, базовое значение интервала шага — 0.</p> <p>Интервалы: ... 3.7 &lt;= x &lt;3.8, <b>3.8</b> &lt;= x &lt;<b>3.9</b>, 3.9 &lt;= x &lt; 4.0...</p>
Round(3.88,5)	Возвращает 5
Round(1.1,1,0.5)	<p>Возвращает 1,5</p> <p>В данном примере значение размера шага — 1, базовое значение интервала шага — 0,5.</p> <p>Интервалы: ...<b>0.5</b> &lt;= x &lt;<b>1.5</b>, 1.5 &lt;= x &lt;2.5, 2.5&lt;= x &lt;3.5...</p>

### 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:** число

#### Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

#### Примеры и результаты:

Примеры	Результаты
sign( 66 )	Возвращает 1
sign( 0 )	Возвращает 0
sign( - 234 )	Возвращает -1



### 5.12 Геопространственные функции

Эти функции используются при работе с геопространственными данными в визуализациях карт.

#### 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, long_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:**

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,0.5) 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:

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 число

#### 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:

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:**

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:**

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:	Геометрия, загруженная как <b>AreaPolygon</b> , преобразуется методом обратного преобразования проекции Меркатора и сохраняется как <b>InvProjectGeometry</b> для использования в визуализациях.

### GeoMakePoint

**GeoMakePoint()** используется в скриптах и выражениях диаграмм для создания и обозначения широты и долготы точки.

#### Синтаксис:

```
GeoMakePoint(lat_field_name, long_field_name)
```

**Возвращаемые типы данных:** строка, с форматированием [долгота, широта]

#### Аргументы:

Аргумент	Описание
lat_field_name	Поле или выражение, относящееся к полю, в котором представлена широта точки.
long_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:

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:

Пример	Результат
В операторе Load:	Проекция Меркатора применяется к геометрии, загруженной в качестве <b>Area</b> , а результат сохраняется в качестве <b>GetProject</b> .

### GeoProjectGeometry

**GeoProjectGeometry()** используется для агрегирования геометрии в область и применения проекции.

**Синтаксис:**

```
GeoProjectGeometry (type, field_name)
```

**Возвращаемые типы данных:** строка

**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:	Геометрия, загруженная как <b>AreaPolygon</b> , преобразуется методом проекции Меркатора и сохраняется как <b>ProjectGeometry</b> 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:

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.
value	<p>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.</p> <div>  <p>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.</p> </div>

**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;
```

### 5.13 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.



*Для большей ясности 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# (страница 563) (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.

---

## 5 Функции в скриптах и выражениях диаграммы

---

```
Interval# (страница 564) (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# (страница 565) (text[, format[, dec_sep[, thou_sep ] ] ])
```

### Num#

**Num#()** преобразует текстовую строку в числовое значение, используя формат, установленный в скрипте загрузки данных или в операционной системе. Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

```
Num# (страница 566) (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# (страница 567) (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# (страница 568) (text[, format])
```

---

### См. также:

р [Функции форматирования \(страница 530\)](#)

### Date#

**Date#** оценивает выражение в качестве даты в формате, указанном во втором аргументе (если указан).

### Синтаксис:

```
Date# (text[, format])
```

**Возвращаемые типы данных:** dual

**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.

Пример	Результаты						
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 <b>StringDate</b> и <b>Date</b> в качестве измерений результаты выглядят следующим образом:  <table><tr><td>StringDate</td><td>Date</td></tr><tr><td>8/7/97</td><td>35649</td></tr><tr><td>8/6/1997</td><td>35648</td></tr></table>	StringDate	Date	8/7/97	35649	8/6/1997	35648
StringDate	Date						
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:** dual

**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.

## 5 Функции в скриптах и выражениях диаграммы

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: .

Пример	Результат
<code>Interval#( A, 'D hh:mm' )</code> , 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:** dual

**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.

## 5 Функции в скриптах и выражениях диаграммы

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

Пример	Результаты	Параметр 1	Параметр 2
Money#(A , '# ##0,00 kr' ) , где A=35 648,37 kr	Строка:	35 648,37 kr	35 648,37 kr
	Число:	35 648,37	3564837
Money#( A, ' \$#', '.', ',' ) , где A= \$35 648,37	Строка:	35 648,37 \$	35 648,37 \$
	Число:	35 648,37	35 648,37

### Num#

**Num#()** преобразует текстовую строку в числовое значение, используя формат, установленный в скрипте загрузки данных или в операционной системе. Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

**Синтаксис:**

```
Num# (text[, format[, dec_sep [, thou_sep ] ] ])
```

**Возвращаемые типы данных:** dual

**Аргументы:**

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая, как будет отформатирована полученная строка даты. Если не указано, используется формат числа, заданный в операционной системе.
dec_sep	Строка, определяющая десятичный разделитель. Если не указано, в скрипте загрузки данных используется набор значений DecimalSep.
thou_sep	Строка, определяющая разделитель тысяч. Если не указано, в скрипте загрузки данных используется набор значений ThousandSep.

Примеры и результаты:

В рассматриваемых ниже примерах предполагается использование двух следующих настроек операционной системы:

## 5 Функции в скриптах и выражениях диаграммы

- Параметр формата числа по умолчанию 1: ###0
- Параметр формата числа по умолчанию 2: #,###0

Пример	Результаты	Параметр 1	Параметр 2
Num#( A, '#.#', ' ','.', ',') где A=35 648,375	Строка:	35 648,375	35 648,375
	Число:	35 648,375	35 648,375

### Text

**Text()** преобразует выражение в текстовый вид даже при возможности обработки его в качестве числа.

#### Синтаксис:

**Text** (expr)

**Возвращаемые типы данных:** dual

Примеры и результаты:

Пример	Результат	
Text( A ) где A=1234	Строка:	1234
	Число:	-
Text( pi( ) )	Строка:	3,1415926535898
	Число:	-

### 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:** dual

#### 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.

## 5 Функции в скриптах и выражениях диаграммы

Примеры и результаты:

В рассматриваемых ниже примерах предполагается использование двух следующих настроек операционной системы:

- Параметр формата времени по умолчанию 1: hh:mm:ss
- Параметр формата времени по умолчанию 2: hh.mm.ss

Пример	Результаты	Параметр 1	Параметр 2
time#( A ) где A=09:00:00	Строка:	09:00:00	09:00:00
	Число:	0,375	-
time#( A, 'hh.mm' ) где A=09.00	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:** dual

**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.

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.



Пример	Результаты								
<p>Добавьте образец скрипта в свое приложение и запустите.</p> <pre>Load *, Timestamp(Timestamp#(String)) as TS; LOAD * INLINE [ String 2015-09-15T12:13:14 1952-10-16T13:14:00+0200 1109-03-01T14:15 ];</pre>	<p>При создании таблицы с помощью <b>String</b> и <b>TS</b> в качестве измерений результаты выглядят следующим образом:</p> <table> <thead> <tr> <th>String</th><th>TS</th></tr> </thead> <tbody> <tr> <td>2015-09-15T12:13:14</td><td>9/15/2015 12:13:14 PM</td></tr> <tr> <td>1952-10-16T13:14:00+0200</td><td>10/16/1952 11:14:00 AM</td></tr> <tr> <td>1109-03-01T14:15</td><td>3/1/1109 2:15:00 PM</td></tr> </tbody> </table>	String	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
String	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								

### 5.14 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 u-значениям в диаграммах или сортировка по столбцам выражений в прямых таблицах не допускается, если в каком-либо из выражений диаграммы используются функции между записями диаграммы. Данные возможности сортировки автоматически отключаются.*

При использовании данных функций автоматически отключается запрещение нулевых значений.

### 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

---

## 5 Функции в скриптах и выражениях диаграммы

---

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 **integer**, которая находит 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.

## 5 Функции в скриптах и выражениях диаграммы

---

**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])

### Межзаписные функции 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()** находит значение поля в таблице для строки, которая уже загружена или существует во встроенной памяти. Можно указать номер строки или таблицы.

**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** (страница 601) (expr)

См. также:

р *Функции над выборкой (страница 619)*

### 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: dual

Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	<p>Если задать значение <b>offset</b> больше 0, можно будет переместить оценку выражения n по строкам выше текущей строки.</p> <p>Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке.</p> <p>Если задать отрицательное число смещения, функция Above будет работать как функция Below с соответствующим положительным числом смещения.</p>
count	<p>Если задать для третьего аргумента <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b>: по одному для каждой строки таблицы элемента <b>count</b>, считая вверх от исходной ячейки.</p> <p>В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. <i>Функции над выборкой (страница 619)</i></p>
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , 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. Межаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.



Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

### Ограничения:

Рекурсивные вызовы возвращают значение NULL.

### Примеры и результаты:

#### Пример 1:

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

Визуализация таблицы для примера 1.

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 **Astrida** показан результат добавления значений `Sum(Sales)` в строки **Betacab** + **Betacab** (539+587). В результате для строки **Betacab** будет показан результат добавления значений `Sum(Sales)` в строки **Canutility** + **Canutility** (683+539).

## 5 Функции в скриптах и выражениях диаграммы

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.*

### 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.

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**.

## 5 Функции в скриптах и выражениях диаграммы

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			<b>2566</b>	-
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

Визуализация таблицы для примера 2.

### Пример 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.

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			<b>2566</b>	-
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

Визуализация таблицы для примера 3.



Пример 4:	Результат								
<p>Функцию Above можно использовать как ввод в функции над выборкой. Например, элемент RangeAvg (Above(Sum(Sales),1,3)).</p>	<p>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.</p> <p>Таблица с элементом Customer в виде измерения выдает следующие результаты для выражения RangeAvg().</p> <table> <tr> <td>Astrida</td><td>-</td></tr> <tr> <td>Betacab</td><td>587</td></tr> <tr> <td>Canutility</td><td>563</td></tr> <tr> <td>Divadip:</td><td>603</td></tr> </table>	Astrida	-	Betacab	587	Canutility	563	Divadip:	603
Astrida	-								
Betacab	587								
Canutility	563								
Divadip:	603								

Данные, используемые в примерах:

```
Monthnames:
LOAD * INLINE [
Month, 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 (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 '|');
```

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

### См. также:

р *Below* — функция диаграммы (страница 578)

р *Bottom* — функция диаграммы (страница 582)

р *Top* — функция диаграммы (страница 602)

р *RangeAvg* (страница 623)

## 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:** dual

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение <b>offset</b> 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 <b>Below</b> будет работать как функция <b>Above</b> с соответствующим положительным числом смещения.
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b> : по одному для каждой строки таблицы элемента <b>count</b> , считая вниз от исходной ячейки. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. <i>Функции над выборкой (страница 619)</i>
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , 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. Межаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.



Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

### Ограничения:

Рекурсивные вызовы возвращают значение NULL.

### Примеры и результаты:

#### Пример 1:

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	-	-	-	-

Визуализация таблицы для примера 1.

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.

## 5 Функции в скриптах и выражениях диаграммы

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**.

## 5 Функции в скриптах и выражениях диаграммы

Пример 2:	Результат								
Функцию <b>Below</b> можно использовать как ввод в функции над выборкой. Например, элемент RangeAvg (Below(Sum(Sales),1,3)).	<p>В аргументах для функции <b>Below()</b> для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения <b>Sum(Sales)</b> 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.</p> <p>Таблица с элементом <b>Customer</b> в виде измерения выдает следующие результаты для выражения RangeAvg().</p>								
	<table><tr><td>Astrida</td><td>659,67</td></tr><tr><td>Betacab</td><td>720</td></tr><tr><td>Canutility</td><td>757</td></tr><tr><td>Divadip:</td><td>-</td></tr></table>	Astrida	659,67	Betacab	720	Canutility	757	Divadip:	-
Astrida	659,67								
Betacab	720								
Canutility	757								
Divadip:	-								

Данные, используемые в примерах:

```
Monthnames:
LOAD * INLINE [
Month, 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 (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 '|');
```

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

### См. также:

р *Above* — функция диаграммы (страница 573)

р *Bottom* — функция диаграммы (страница 582)

р *Top* — функция диаграммы (страница 602)

р *RangeAvg* (страница 623)

## 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:** dual

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать для <b>offset</b> значение n больше 1, можно будет переместить оценку выражения n по строкам выше нижней строки.  Если задать отрицательное число смещения, функция <b>Bottom</b> будет работать как функция <b>Top</b> с соответствующим положительным числом смещения.
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет не одно, а ряд значений элемента <b>count</b> : по одному для каждой последней строки элемента <b>count</b> текущего сегмента столбца. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. <i>Функции над выборкой (страница 619)</i>
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.



Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор **TOTAL**, выражение оценивается по всей таблице.



Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

### Ограничения:

Рекурсивные вызовы возвращают значение NULL.

### Примеры и результаты:

#### Пример: 1

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

Визуализация таблицы для примера 1.

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.)

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

*Первая таблица для примера 2. Значение элемента Bottom для меры End value основано на элементах Month (Dec).*



## 5 Функции в скриптах и выражениях диаграммы

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

Вторая таблица для примера 2. Значение элемента *Bottom* для меры *End value* основано на элементе *Product* (BB для Astrida).

Дополнительную информацию см. в примере 2 для функции **Above**.

Пример: 3	Результат								
Функцию <b>Bottom</b> можно использовать как ввод в функции над выборкой. Например, элемент <code>RangeAvg (Bottom(Sum(Sales),1,3))</code> .	<p>В аргументах для функции <b>Bottom()</b> для элемента <code>offset</code> задано значение 1, а для элемента <code>count</code> задано значение 3. Функция находит результаты выражения <b>Sum(Sales)</b> в трех строках, начиная со строки над нижней строкой в сегменте столбца (поскольку <code>offset=1</code>) и в двух строках над ней (если есть строка). Эти три значения используются как ввод в функцию <code>RangeAvg()</code>, которая находит среднее значение в предоставленном диапазоне чисел.</p> <p>Таблица с элементом <b>Customer</b> в виде измерения выдает следующие результаты для выражения <code>RangeAvg()</code>.</p>								
	<table><tr><td>Astrida</td><td>659,67</td></tr><tr><td>Betacab</td><td>659,67</td></tr><tr><td>Canutility</td><td>659,67</td></tr><tr><td>Divadip:</td><td>659,67</td></tr></table>	Astrida	659,67	Betacab	659,67	Canutility	659,67	Divadip:	659,67
Astrida	659,67								
Betacab	659,67								
Canutility	659,67								
Divadip:	659,67								

Monthnames:  
LOAD \* INLINE [  
Month, 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 (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 '|');
```

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

**См. также:**

*p Top — функция диаграммы (страница 602)*

### 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:** dual

**Аргументы:**

Аргумент	Описание
ColumnNo	Номер столба в таблице, содержащей меру. <div> Функция <i>Column()</i> игнорирует столбцы измерений.</div>

**Ограничения:**

Если элемент **ColumnNo** ссылается на столбец, для которого нет мер, возвращается значение NULL.

## 5 Функции в скриптах и выражениях диаграммы

Рекурсивные вызовы возвращают значение NULL.

### Примеры и результаты:

#### Пример: Процентное соотношение итоговых продаж

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
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	Результаты
Элемент Order Value добавляется к таблице в качестве меры с выражением: <code>sum(UnitPrice*UnitSales)</code> .	Результат элемента Column(1) взят из столбца Order Value, поскольку это первый столбец с мерами.
Элемент Total Sales Value добавляется как мера с выражением: <code>sum(TOTAL UnitPrice*UnitSales)</code>	Результат элемента Column(2) взят из столбца Total Sales Value, поскольку это второй столбец с мерами.
Элемент % Sales добавляется как мера с выражением <code>100*column(1)/column(2)</code>	См. результат в столбце % Sales в примере <i>Процентное соотношение итоговых продаж (страница 587)</i> .

Примеры	Результаты
Выполнить выборку Customer A.	Выборка изменяет элемент Total Sales Value и, следовательно, элемент %Sales. См. пример <i>Процентное соотношение продаж для выбранного клиента (страница 587)</i> .

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. Возвратится число измерений во всех строках, за исключением итогового числа, которое будет равно 0. Для всех типов диаграмм, кроме сводной таблицы, будет возвращено количество измерений во всех строках, кроме общей, которая будет равна 0.

#### Пример:

Обычно функция dimensionality используется, когда необходимо выполнить вычисление, только при наличии значения для измерения.

Пример	Результат
Для таблицы, содержащей измерение UnitSales, возможно, будет необходимо только указать, что накладная отправлена:  <code>IF(Dimensionality()=3, "Invoiced").</code>	

### 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**.

#### Синтаксис:

```
Exists (field_name [, expr])
```

**Возвращаемые типы данных:** Boolean

#### Аргументы:

Аргумент	Описание
field_name	Имя выражения строки, которое оценивает имя поля, поиск которого будет выполняться. Поле должно существовать в данных, загруженных скриптом до настоящего времени.
expr	Выражение, которое оценивает значение поля для поиска в поле, указанном в <b>field-name</b> . При его отсутствии принимается значение текущей записи в указанном поле.

#### Примеры и результаты:

Пример	Результат
<code>Exists (Employee)</code>	Возвращает -1 (True), если значение поля <b>Employee</b> в текущей записи уже существует в любой ранее прочитанной записи, содержащей это поле.
<code>Exists(Employee, 'Bill')</code>	Возвращает -1 (True), если значение поля <b>'Bill'</b> найдено в текущем содержимом поля <b>Employee</b> .  Операторы <code>Exists (Employee, Employee)</code> и <code>Exists (Employee)</code> эквивалентны.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат								
<p>Employees:  LOAD * inline [  Employee ID Salary  Bill 001 20000  John 002 30000  Steve 003 35000  ] (delimiter is ' ');</p> <p>Citizens:  Load * inline [  Name Address  Bill New York  Mary London  Steve Chicago  Lucy Paris  John Miami  ] (delimiter is ' ');</p> <p>EmployeeAddresses:  Load Name as Employee, Address Resident  Citizens where Exists (Employee, Name);</p> <p>Drop Tables Employees, Citizens;</p>	<p>В результате в модели данных будет получена таблица с именем EmployeeAddresses, которую можно просмотреть как визуализацию таблицы с помощью измерений Employee и Address.</p> <p>Предложение where: where Exists (Employee, Name), означает только имена из таблицы Citizens, загруженные в новую таблицу, которые также находятся в таблице Employees. Оператор Drop удаляет временные таблицы Employees и Citizens во избежание неопределенности.</p> <table> <thead> <tr> <th>Employee</th><th>Address</th></tr> </thead> <tbody> <tr> <td>Bill</td><td>New York</td></tr> <tr> <td>John</td><td>Miami</td></tr> <tr> <td>Steve</td><td>Chicago</td></tr> </tbody> </table>	Employee	Address	Bill	New York	John	Miami	Steve	Chicago
Employee	Address								
Bill	New York								
John	Miami								
Steve	Chicago								
<p>Замена оператора в данных образца в предыдущем примере, который создает таблицу EmployeeAddresses с помощью where not Exists следующим образом.</p> <p>NonEmployee:  Load Name as Employee, Address Resident  Citizens where not Exists (Employee, Name);</p>	<p>Предложение where, включая not: where not Exists (Employee, Name), означает только имена из таблицы Citizens, загруженные в новую таблицу, которые отсутствуют в таблице Employees.</p> <table> <thead> <tr> <th>Employee</th><th>Address</th></tr> </thead> <tbody> <tr> <td>Mary</td><td>London</td></tr> <tr> <td>Lucy</td><td>Paris</td></tr> </tbody> </table>	Employee	Address	Mary	London	Lucy	Paris		
Employee	Address								
Mary	London								
Lucy	Paris								

Данные, используемые в примере:

Employees:  
LOAD \* inline [  
Employee|ID|Salary  
Bill|001|20000  
John|002|30000  
Steve|003|35000  
] (delimiter is '|');

Citizens:  
Load \* inline [  
Name|Address  
Bill|New York  
Mary|London  
Steve|Chicago  
Lucy|Paris  
John|Miami  
] (delimiter is '|');

## 5 Функции в скриптах и выражениях диаграммы

EmployeeAddresses:

Load Name as Employee, Address Resident Citizens where Exists (Employee, Name);

Drop Tables Employees, Citizens;

### FieldIndex

Функция **FieldIndex()** возвращает позицию значения поля **value** в поле **field\_name** (в порядке загрузки).

#### Синтаксис:

```
FieldIndex (field_name , value)
```

**Возвращаемые типы данных:** целое число

#### 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 <b>field_name</b> .

#### Ограничения:

Если элемент **value** не может быть найден среди значений поля **field\_name**, 0 возвращается.

#### Примеры и результаты:

В следующих примерах используется поле: **First name** из таблицы **Names**.

Примеры	Результаты
Добавьте образец данных в свое приложение и запустите.	Таблица <b>Names</b> 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 <b>First name</b> Обратите внимание, что в фильтре элемент <b>John</b> появится как число 2 сверху, поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
<code>FieldIndex ('First name','Peter')</code>	4, поскольку элемент <b>FieldIndex()</b> возвращает только одно значение, которое встречается сначала в порядке загрузки.
Функция скрипта. При условии, что таблица <b>Names</b> загружается как в данных примера:	
<code>John1: Load FieldIndex('First name','John') as MyJohnPos Resident Names;</code>	myJohnPos=1, поскольку элемент «John» появляется сначала в порядке загрузки поля <b>First name</b> . Обратите внимание, что в фильтре элемент <b>John</b> появится как число 2 сверху, поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.
<code>Peter1: Load FieldIndex('First name','Peter') as MyPeterPos Resident Names;</code>	myPeterPos=4, поскольку элемент <b>FieldIndex()</b> 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:** dual

**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.

**Примеры и результаты:**

В следующих примерах используется поле: **First name** из таблицы **Names**.

Примеры	Результаты
Добавьте образец данных в свое приложение и запустите.	Таблица <b>Names</b> загружается как в данных для образца.
Функция диаграммы. В таблице, содержащей измерение First name, добавьте следующую меру:	
Fieldvalue('First name','1')	John, поскольку элемент John появляется сначала в порядке загрузки поля <b>First name</b> . Обратите внимание, что в фильтре элемент <b>John</b> появится как число 2 сверху после элемента <b>Jane</b> , поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.
Fieldvalue('First name','7')	Значение NULL, поскольку в поле <b>First name</b> только 6 значений.
Функция скрипта. При условии, что таблица <b>Names</b> загружается как в данных примера:	
John1: Load Fieldvalue('First name',1) as MyPos1 Resident Names;	myPos1=John, поскольку элемент 'John' появляется сначала в порядке загрузки поля <b>First name</b> .

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
Peter1: Load FieldValue('First name',7) as MyPos2 Resident Names;	MyPos2= — (Null), поскольку в поле <b>First name</b> только 6 значений.

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 FieldValue('First name',1) as MyPos1
Resident Names;
```

```
Peter1:
Load FieldValue('First name',7) as MyPos2
Resident Names;
```

### FieldValueCount

Функция **FieldValueCount()** — это функция **integer**, которая находит уникальные значения в поле.

**Синтаксис:**

```
FieldValueCount (field_name)
```

**Возвращаемые типы данных:** целое число

**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	Результаты
Добавьте образец данных в свое приложение и запустите.	Таблица <b>Names</b> загружается как в данных для образца.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
Функция диаграммы. В таблице, содержащей измерение First name, добавьте следующую меру:	
FieldvalueCount('First name')	Значение 5, поскольку элемент <b>Peter</b> появляется дважды.
FieldvalueCount('Initials')	Значение 6, поскольку элемент <b>Initials</b> имеет только уникальные значения.
Функция скрипта. При условии, что таблица <b>Names</b> загружается как в данных примера:	
John1: Load FieldvalueCount('First name') as MyFieldCount1 Resident Names;	MyFieldCount1=5, поскольку элемент 'John' появляется дважды.
John1: Load FieldvalueCount('Initials') as MyInitialsCount1 Resident Names;	MyFieldCount1=6, поскольку элемент 'Initials' имеет только уникальные значения.

Данные, используемые в примере:

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:** dual

**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 <b>match_field_value</b> . Вводимое значение необходимо задать в виде строки (например литералы ссылочного типа).
match_field_value	Значение, которое необходимо искать в поле <b>match_field_name</b> .
table_name	Имя таблицы, в которой необходимо искать значение. Вводимое значение необходимо задать в виде строки (например, литералы ссылочного типа).  Если элемент <b>table_name</b> 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.

Примеры и результаты:

Пример	Результат																														
<p>Данные образца используют функцию <b>Lookup()</b> в следующем виде:</p> <pre>Lookup('Category', 'ProductID', ProductID, 'ProductList')</pre> <p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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</pre>	<p>Сначала загружается таблица <b>ProductList</b>.</p> <p>Функция <b>Lookup()</b> используется для построения таблицы <b>OrderData</b>. Она указывает третий аргумент как <b>ProductID</b>. Это поле, для которого будет выполняться поиск значения во втором аргументе <b>'ProductID'</b> в таблице <b>ProductList</b>, как определено завершающими одинарными кавычками.</p> <p>Функция возвращает значение для <b>'Category'</b> (в таблице <b>ProductList</b>), загруженной как <b>CategoryID</b>.</p> <p>Оператор <b>drop</b> удаляет таблицу <b>ProductList</b> из модели данных, поскольку она не требуется. Результаты таблицы <b>OrderData</b> будут следующими:</p> <table><tr><th>ProductID</th><th>InvoiceID</th><th>CustomerID</th><th>Units</th><th>CategoryID</th></tr><tr><td>1</td><td>1</td><td>Astrida</td><td>8</td><td>1</td></tr><tr><td>2</td><td>1</td><td>Astrida</td><td>6</td><td>1</td></tr><tr><td>3</td><td>2</td><td>Betacab</td><td>10</td><td>2</td></tr><tr><td>3</td><td>3</td><td>Divadip</td><td>5</td><td>2</td></tr><tr><td>4</td><td>4</td><td>Divadip</td><td>10</td><td>3</td></tr></table>	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
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																											



*Функция **Lookup()** гибкая, она может получить доступ к любой ранее загруженной таблице. Тем не менее, она медленно сравнивается с функцией **ApplyMap()**.*

**См. также:**

р **ApplyMap** (страница 613)

### 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.

## 5 Функции в скриптах и выражениях диаграммы

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

### Синтаксис:

```
NoOfRows ( [TOTAL] )
```

**Возвращаемые типы данных:** целое число

### Аргументы:

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.

### Пример:

```
if( RowNo( )= NoOfRows( ), 0, Above( sum( Sales )))
```

### См. также:

*p RowNo* — функция диаграммы (страница 388)

## Peek

Функция **Peek()** находит значение поля в таблице для строки, которая уже загружена или существует во встроенной памяти. Можно указать номер строки или таблицы.

### Синтаксис:

```
Peek (field_name[, row_no[, table_name ] ])
```

**Возвращаемые типы данных:** dual

### 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 <b>row</b> не задан, используется -1.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
table_ name	Метка таблицы без двоеточия на конце. Если элемент <b>table_name</b> не указан, принимается текущая таблица. При использовании вне оператора <b>LOAD</b> или относительно другой таблицы должен быть включен элемент <b>table_name</b> .


### Ограничения:

В первой записи внутренней таблицы функция возвращает значение NULL.

### Примеры и результаты:

Пример	Результат														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ' ');  FirstEmployee: Load EmployeeCode, Peek(EmployeeCode,0) As EmpCode Resident EmployeeDates;</pre>	<p>EmpCode = 101, поскольку Peek(EmployeeCode,0) возвращает первое значение элемента EmployeeCode в таблице EmployeeDates.</p> <p>Замена значения аргумента <b>row_no</b> возвращает значения других строк в таблице следующим образом:</p> <p>Peek(EmployeeCode,2) возвращает третье значение в таблице: 102.</p> <p>Тем не менее, обратите внимание, что без указания таблицы в качестве третьего аргумента <b>table_no</b> функция ссылается на текущую (в данном случае внутреннюю) таблицу. Результатом Peek(EmployeeCode,-2) будет множество значений:</p> <table><thead><tr><th>EmployeeCode</th><th>EmpCode</th></tr></thead><tbody><tr><td>101</td><td>-</td></tr><tr><td>102</td><td>-</td></tr><tr><td>103</td><td>101</td></tr><tr><td>104</td><td>102</td></tr><tr><td>105</td><td>103</td></tr><tr><td>106</td><td>104</td></tr></tbody></table>	EmployeeCode	EmpCode	101	-	102	-	103	101	104	102	105	103	106	104
EmployeeCode	EmpCode														
101	-														
102	-														
103	101														
104	102														
105	103														
106	104														
<pre>FirstEmployee: Load EmployeeCode, Peek(EmployeeCode,- 2,'EmployeeDates') As EmpCode Resident EmployeeDates;</pre>	<p>При указании аргумента <b>table_no</b> как 'EmployeeDates' функция возвращает предпоследнее значение элемента EmployeeCode в таблице EmployeeDates: 105.</p>														

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат																																				
<p>Функцию <b>Peek()</b> можно использовать для указания ссылки на данные, которые еще не загружены.</p> <p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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)&amp;', '&amp;Value,Value) AS List RESIDENT T1 ORDER BY ID ASC; DROP TABLE T1;</pre>	<p>Создайте таблицу на листе в вашем приложении с элементами <b>ID</b>, <b>List</b> и <b>Value</b> в качестве измерений.</p> <table><tr><th>ID</th><th>List</th><th>Value</th></tr><tr><td>1</td><td>6</td><td>6</td></tr><tr><td>1</td><td>6,3</td><td>3</td></tr><tr><td>1</td><td>6,3,4</td><td>4</td></tr><tr><td>2</td><td>11</td><td>11</td></tr><tr><td>2</td><td>11,10</td><td>10</td></tr><tr><td>2</td><td>11,10,1</td><td>1</td></tr><tr><td>3</td><td>8</td><td>8</td></tr><tr><td>3</td><td>8,7</td><td>7</td></tr><tr><td>5</td><td>13</td><td>13</td></tr><tr><td>5</td><td>13,2</td><td>2</td></tr><tr><td>5</td><td>13,2,78</td><td>78</td></tr></table> <p>Оператор <b>IF()</b> строится на основе временной таблицы T1.</p> <p>peek(ID) ссылается на поле ID в предыдущей строке в текущей таблице T2.</p> <p>peek(List) ссылается на поле List в предыдущей строке в таблице T2, которая строится в настоящее время как оценивающееся выражение.</p> <p>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.</p> <p>Если функция Peek(List) уже содержит объединенный результат, новый результат элемента Peek(List) будет объединен с ним.</p> <div><div></div><div><p>Обратите внимание на предложение <b>Order by</b>. Оно указывает порядок организации таблицы (по ID по возрастанию). Без этого функция Peek() будет использовать тот обязательный</p></div></div>	ID	List	Value	1	6	6	1	6,3	3	1	6,3,4	4	2	11	11	2	11,10	10	2	11,10,1	1	3	8	8	3	8,7	7	5	13	13	5	13,2	2	5	13,2,78	78
ID	List	Value																																			
1	6	6																																			
1	6,3	3																																			
1	6,3,4	4																																			
2	11	11																																			
2	11,10	10																																			
2	11,10,1	1																																			
3	8	8																																			
3	8,7	7																																			
5	13	13																																			
5	13,2	2																																			
5	13,2,78	78																																			
Синтаксис скрипта и функции диаграммы - Qlik Sense, 3.2	<div><div><p>порядок, который указан во внутренней таблице, что может привести к непредсказуемым результатам.</p></div><div>600</div></div>																																				



### 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:** dual

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения. Выражение может содержать вложенные функции <b>previous()</b> , 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.

#### 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.

Пример	Результат																										
<pre>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 ' ');</pre>	<p>При использовании функции <b>Previous()</b> в операторе <b>Load</b> можно сравнить текущее значение элемента Sales с предшествующим значением и использовать его в третьем поле Increase.</p> <table> <thead> <tr> <th>Month</th><th>Increase</th></tr> </thead> <tbody> <tr><td>1</td><td>-</td></tr> <tr><td>2</td><td>1</td></tr> <tr><td>3</td><td>2</td></tr> <tr><td>4</td><td>2</td></tr> <tr><td>5</td><td>4</td></tr> <tr><td>6</td><td>0</td></tr> <tr><td>7</td><td>1</td></tr> <tr><td>8</td><td>1</td></tr> <tr><td>9</td><td>9</td></tr> <tr><td>10</td><td>3</td></tr> <tr><td>11</td><td>5</td></tr> <tr><td>12</td><td>1</td></tr> </tbody> </table>	Month	Increase	1	-	2	1	3	2	4	2	5	4	6	0	7	1	8	1	9	9	10	3	11	5	12	1
Month	Increase																										
1	-																										
2	1																										
3	2																										
4	2																										
5	4																										
6	0																										
7	1																										
8	1																										
9	9																										
10	3																										
11	5																										
12	1																										

### Тор — функция диаграммы

Функция **Тор()** 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 **Тор()** используется для оценки в первой строке текущего столбца в эквиваленте прямой таблицы диаграммы.

#### Синтаксис:

```
Тор ([TOTAL] expr [ , offset [,count ]])
```

**Возвращаемые типы данных:** dual

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
offset	<p>Если задать значение <b>offset</b> элемента больше 1, можно будет переместить оценку выражения n по строкам ниже верхней строки.</p> <p>Если задать отрицательное число смещения, функция <b>Top</b> будет работать как функция <b>Bottom</b> с соответствующим положительным числом смещения.</p>
count	<p>Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет ряд значений элемента <b>count</b>: по одному для каждой последней строки элемента <b>count</b> текущего сегмента столбца. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. <i>Функции над выборкой (страница 619)</i></p>
TOTAL	<p>Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b>, текущий сегмент столбца всегда равен всему столбцу.</p>



Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор **TOTAL**, выражение оценивается по всей таблице.



Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

### Ограничения:

Рекурсивные вызовы возвращают значение NULL.

### Примеры и результаты:

#### Пример: 1

Top and Bottom					
Customer	Q	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

## 5 Функции в скриптах и выражениях диаграммы

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: 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.

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

Первая таблица для примера 2. Значение элемента **Top** для меры **First value** основано на элементах **Month** (**Jan**).

## 5 Функции в скриптах и выражениях диаграммы

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

Вторая таблица для примера 2. Значение элемента Top для меры First value основано на элементе Product (AA для Astrida).

Дополнительную информацию см. в примере 2 для функции **Above**.

Пример: 3	Результат								
Функцию <b>Top</b> можно использовать как ввод в функции над выборкой. Например, элемент RangeAvg (Top(Sum(Sales),1,3)).	В аргументах для функции <b>Top()</b> для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения <b>Sum(Sales)</b> 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 <b>Customer</b> в виде измерения выдает следующие результаты для выражения RangeAvg().								
	<table><tr><td>Astrida</td><td>603</td></tr><tr><td>Betacab</td><td>603</td></tr><tr><td>Canutility</td><td>603</td></tr><tr><td>Divadip:</td><td>603</td></tr></table>	Astrida	603	Betacab	603	Canutility	603	Divadip:	603
Astrida	603								
Betacab	603								
Canutility	603								
Divadip:	603								

Monthnames:  
LOAD \* INLINE [  
Month, 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 (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 '|');
```

Чтобы выполнить сортировку месяцев в правильном порядке, при создании визуализаций перейдите в раздел **Sorting** на панели свойств, выберите элемент **Month** и установите флажок **Sort by expression**. В поле выражения напишите monthnumber.

---

### См. также:

р *Bottom* — функция диаграммы (страница 582)  
р *Above* — функция диаграммы (страница 573)  
р *Sum* — функция диаграммы (страница 193)  
р *RangeAvg* (страница 623)  
р *Функции над выборкой* (страница 619)

## 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.

## After — функция диаграммы

Функция **After()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце после текущего столбца в сегменте строки сводной таблицы.

### Синтаксис:

```
after ( [TOTAL] expr [, offset [, count ]])
```



Данная функция возвращает значение *NULL* во всех типах диаграмм, кроме сводных таблиц.

### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	<p>Если задать значение <b>offset</b> n больше 1, можно переместить оценку выражения на n строк вправо от текущей строки.</p> <p>Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке.</p> <p>Если задать отрицательное число смещения, функция <b>After</b> будет работать как функция <b>Before</b> с соответствующим положительным числом смещения.</p>
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b> , 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 <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.

В последнем столбце сегмента строки будет возвращено значение *NULL*, так как после этого столбца нет других столбцов.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки. Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

### Пример:

```
after( sum( sales ))  
after( sum( sales ), 2 )  
after( total sum( sales ))  
rangeavg (after(sum(x),1,3)) возвращает средний из трех результатов функции sum(x), оцененной в  
трех столбцах непосредственно справа от текущего столбца.
```

### Before — функция диаграммы

Функция **Before()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце перед текущим столбцом в сегменте строки сводной таблицы.

#### Синтаксис:

```
before ([TOTAL] expr [, offset [, count]])
```



Данная функция возвращает значение *NULL* во всех типах диаграмм, кроме сводных таблиц.

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	<p>Если задать значение <b>offset</b> <i>n</i> больше 1, можно переместить оценку выражения на <i>n</i> строк влево от текущей строки.</p> <p>Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке.</p> <p>Если задать отрицательное число смещения, функция <b>Before</b> будет работать как функция <b>After</b> с соответствующим положительным числом смещения.</p>
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b> , 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 <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.

В первом столбце сегмента строки будет возвращено значение *NULL*, так как перед этим столбцом нет других столбцов.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки. Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

#### Примеры:

```
before( sum( Sales ))  
before( sum( Sales ), 2 )  
before( total sum( Sales ))
```



`rangeavg (before(sum(x),1,3))` возвращает средний из трех результатов функции **sum(x)**, оцененной в трех столбцах непосредственно слева от текущего столбца.

### First — функция диаграммы

Функция **First()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в первом столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

#### Синтаксис:

```
first([TOTAL] expr [, offset [, count]])
```

#### Аргументы:

Аргумент	Описание
expression	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение <b>offset</b> 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 <b>First</b> будет работать как функция <b>Last</b> с соответствующим положительным числом смещения.
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b> , 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 <b>TOTAL</b> , 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 межполевым порядке сортировки. Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

#### Примеры:

```
first( sum( sales ))  
first( sum( sales ), 2 )  
first( total sum( sales )  
rangeavg (first(sum(x),1,5)) возвращает средний из результатов функции sum(x),  
оцененной в пяти самых левых столбцах текущего сегмента строки.
```

### Last — функция диаграммы

Функция **Last()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в последнем столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

#### Синтаксис:

```
last([TOTAL] expr [, offset [, count]])
```

#### Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение <b>offset</b> 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 <b>First</b> будет работать как функция <b>Last</b> с соответствующим положительным числом смещения.
count	Если задать для третьего параметра <b>count</b> значение больше 1, функция вернет диапазон значений элемента <b>count</b> , 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 <b>TOTAL</b> , 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 межполевого порядка сортировки. Межполевого 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:

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , текущий сегмент столбца всегда равен всему столбцу.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки. Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

### Пример:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

## NoOfColumns — функция диаграммы

Функция **NoOfColumns()** возвращает количество столбцов в текущем сегменте строки сводной таблицы.

### Синтаксис:

```
NoOfColumns ([total])
```

### Аргументы:

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс <b>TOTAL</b> , 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 )))
```

### 5.15 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:

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

### 5.16 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:** dual

##### Аргументы:

Аргумент	Описание
map_name	Имя таблицы сопоставления, созданной ранее с помощью операторов <b>mapping load</b> или <b>mapping select</b> . Имя таблицы должно быть заключено в одинарные прямые кавычки.
expression	Выражение, результат которого должен быть сопоставлен.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
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.

### 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:

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:

Аргумент	Описание
map_name	Имя таблицы сопоставления, считанной ранее оператором <b>mapping load</b> или <b>mapping select</b> . Имя должно быть заключено в одинарные прямые кавычки.
expression	Выражение, результат которого должен быть сопоставлен по подстрокам.

#### Пример:

В этом примере мы загружаем список моделей продукта. Каждая модель имеет набора 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:

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

### 5.17 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  $\pi$  (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 ( )
```

## 5.18 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.

#### Null

Функция **Null** возвращает значение NULL.

```
NULL ( )
```

#### IsNull

Функция **IsNull** проверяет, является ли значение выражения NULL. Если да, то функция возвращает -1 (True), в противном случае — 0 (False).

```
IsNull (expr )
```

#### 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 -.

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 -.

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

## 5.19 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 приравниваются к 0 в отличие от оператора +.

```
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:**

р *Функции между записями (страница 569)*

### RangeAvg

**RangeAvg()** возвращает среднее значение диапазона. Для ввода в функцию может использоваться диапазон значений или выражение.

#### Синтаксис:

```
RangeAvg (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

#### 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.

#### Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

#### Примеры и результаты:

Примеры	Результаты
RangeAvg (1,2,4)	Возвращает 2,33333333
RangeAvg (1, 'xyz')	Возвращает 1
RangeAvg (null( ), 'abc')	Возвращает NULL

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица показывает возвращенные значения функции <b>MyRangeAvg</b> для каждой записи в таблице.</p> <table> <tr> <th>RangeID</th><th>MyRangeAvg</th></tr> <tr> <td>1</td><td>7</td></tr> <tr> <td>2</td><td>4</td></tr> <tr> <td>3</td><td>6</td></tr> <tr> <td>4</td><td>12.666</td></tr> <tr> <td>5</td><td>6.333</td></tr> <tr> <td>6</td><td>5</td></tr> </table>	RangeID	MyRangeAvg	1	7	2	4	3	6	4	12.666	5	6.333	6	5
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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

MyField	RangeAvg (Above (MyField,0,3))	
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	9,3333333333	
5	10. 3333333333	
9	10,6666666667	



```
RangeTab:
LOAD * INLINE [
myField
10
2
8
18
5
9
] ;
```

### См. также:

p Avg — функция диаграммы (страница 233)

p Count — функция диаграммы (страница 198)

## 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:** число

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:

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 <b>x-value</b> или диапазону значений <b>y-values</b> .
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	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.										
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre> 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 ' ');</pre>	<p>В таблице с измерением ID1 и следующей мерой: RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) функция <b>RangeCorrel()</b> приобретает значение <b>Correl</b>, находящееся в диапазоне шести повторяемых значений x,y для каждого из значений ID1.</p> <table> <thead> <tr> <th>ID1</th><th>MyRangeCorrel</th></tr> </thead> <tbody> <tr> <td>01</td><td>-0.9517</td></tr> <tr> <td>02</td><td>-0.5209</td></tr> <tr> <td>03</td><td>-0.5209</td></tr> <tr> <td>04</td><td>-0.1599</td></tr> </tbody> </table>	ID1	MyRangeCorrel	01	-0.9517	02	-0.5209	03	-0.5209	04	-0.1599
ID1	MyRangeCorrel										
01	-0.9517										
02	-0.5209										
03	-0.5209										
04	-0.1599										

Примеры	Результаты										
<pre> XY: LOAD recno() as RangeID, * Inline [ X Y 2 3 6 8 9 4 8 5 ](delimiter is ' '); </pre>	<p>В таблице с измерением RangeID и следующей мерой: RangeCorrel(Below(X,0,4,BelowY,0,4)) функция <b>RangeCorrel()</b> использует результаты функций <b>Below()</b>, которые на основе третьего аргумента (count) установлены на значение 4. Функция также формирует диапазон из четырех значений x-y из загруженной таблицы XY.</p> <table> <thead> <tr> <th>RangeID</th><th>MyRangeCorrel2</th></tr> </thead> <tbody> <tr> <td>01</td><td>0.2492</td></tr> <tr> <td>02</td><td>-0.9959</td></tr> <tr> <td>03</td><td>-1.0000</td></tr> <tr> <td>04</td><td>-</td></tr> </tbody> </table> <p>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.</p>	RangeID	MyRangeCorrel2	01	0.2492	02	-0.9959	03	-1.0000	04	-
RangeID	MyRangeCorrel2										
01	0.2492										
02	-0.9959										
03	-1.0000										
04	-										

**См. также:**

р *Correl* — функция диаграммы (страница 236)

## RangeCount

**RangeCount()** возвращает количество текстовых и числовых значений в выражении или поле.

**Синтаксис:**

```
RangeCount (first_expr[, Expression])
```

**Возвращаемые типы данных:** целое число

**Аргументы:**

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

## 5 Функции в скриптах и выражениях диаграммы

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	Результаты														
RangeCount (1,2,4)	Возвращает 3														
RangeCount (2,'xyz')	Возвращает 2														
RangeCount (null( ))	Возвращает 0														
RangeCount (2,'xyz', null())	Возвращает 2														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица показывает возвращенные значения функции MyRangeCount для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>MyRangeCount</th></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>3</td></tr><tr><td>3</td><td>3</td></tr><tr><td>4</td><td>3</td></tr><tr><td>5</td><td>3</td></tr><tr><td>6</td><td>3</td></tr></table>	RangeID	MyRangeCount	1	3	2	3	3	3	4	3	5	3	6	3
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 второго и третьего аргумента функции **Above()** значение и 3, будут возвращены значения из трех полей над текущей строкой, если строк достаточно, которые принимаются как значения, вводимые в функцию **RangeSum()**.

Данные, используемые в примерах:

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 (страница 198)

### 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:** число

**Аргументы:**

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

## 5 Функции в скриптах и выражениях диаграммы

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.														
Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.  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.  <table><thead><tr><th>RangeID</th><th>MyRangeFrac</th></tr></thead><tbody><tr><td>1</td><td>6</td></tr><tr><td>2</td><td>3</td></tr><tr><td>3</td><td>8</td></tr><tr><td>4</td><td>11</td></tr><tr><td>5</td><td>5</td></tr><tr><td>6</td><td>4</td></tr></tbody></table>	RangeID	MyRangeFrac	1	6	2	3	3	8	4	11	5	5	6	4
RangeID	MyRangeFrac														
1	6														
2	3														
3	8														
4	11														
5	5														
6	4														

### Примеры и результаты:

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 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

## 5 Функции в скриптах и выражениях диаграммы

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.

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1,5
3	2
4	3
5	4
6	5

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
myField
1
2
3
4
5
6
] ;
```

### См. также:

р *Above* — функция диаграммы (страница 573)

р *Fractile* — функция диаграммы (страница 239)

## 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:** число

**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.

Примеры	Результаты														
RangeIRR(-70000,12000,15000,18000,21000,26000)	Возвращает 0,0866														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ' ');</pre>	<p>Результирующая таблица показывает возвращенные значения функции RangeIRR для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>RangeIRR</th></tr><tr><td>1</td><td>0.0639</td></tr><tr><td>2</td><td>0.8708</td></tr><tr><td>3</td><td>-</td></tr><tr><td>4</td><td>5.8419</td></tr><tr><td>5</td><td>0.9318</td></tr><tr><td>6</td><td>-0.2566</td></tr></table>	RangeID	RangeIRR	1	0.0639	2	0.8708	3	-	4	5.8419	5	0.9318	6	-0.2566
RangeID	RangeIRR														
1	0.0639														
2	0.8708														
3	-														
4	5.8419														
5	0.9318														
6	-0.2566														

**См. также:**

р *Функции между записями (страница 569)*

## RangeKurtosis

**RangeKurtosis()** возвращает значение, соответствующее эксцессу диапазона чисел.



### Синтаксис:

```
RangeKurtosis (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

### 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.

### Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

### Примеры и результаты:

Примеры	Результаты
RangeKurtosis (1,2,4,7)	Возвращает -0,28571428571429

### См. также:

*p Kurtosis* — функция диаграммы (страница 243)

## RangeMax

**RangeMax()** возвращает самые высокие числовые значения, обнаруженные в выражении или поле.

### Синтаксис:

```
RangeMax (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

### 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	Результаты														
<code>RangeMax (1,2,4)</code>	Возвращает 4														
<code>RangeMax (1,'xyz')</code>	Возвращает 1														
<code>RangeMax (null( ), 'abc')</code>	Возвращает NULL														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица показывает возвращенные значения функции <b>MyRangeMax</b> для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>MyRangeMax</th></tr><tr><td>1</td><td>10</td></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>8</td></tr><tr><td>4</td><td>18</td></tr><tr><td>5</td><td>9</td></tr><tr><td>6</td><td>9</td></tr></table>	RangeID	MyRangeMax	1	10	2	7	3	8	4	18	5	9	6	9
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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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.

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
<code>RangeMaxString (1,2,4)</code>	Возвращает 4
<code>RangeMaxString ('xyz','abc')</code>	Возвращает «xyz»
<code>RangeMaxString (5,'abc')</code>	Возвращает «abc»
<code>RangeMaxString (null( ))</code>	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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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 (страница 357)

### RangeMin

**RangeMin()** возвращает самые низкие числовые значения, обнаруженные в выражении или поле.

**Синтаксис:**

```
RangeMin (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

**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	Результаты
<code>RangeMin (1,2,4)</code>	Возвращает 1
<code>RangeMin (1,'xyz')</code>	Возвращает 1
<code>RangeMin (null( ), 'abc')</code>	Возвращает NULL

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица показывает возвращенные значения функции <b>MyRangeMin</b> для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>MyRangeMin</th></tr><tr><td>1</td><td>5</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>2</td></tr><tr><td>4</td><td>9</td></tr><tr><td>5</td><td>5</td></tr><tr><td>6</td><td>2</td></tr></table>	RangeID	MyRangeMin	1	5	2	2	3	2	4	9	5	5	6	2
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:

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
1 ;
```

**См. также:**

*p Min — функция диаграммы (страница 184)*

### 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.

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
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.

## 5 Функции в скриптах и выражениях диаграммы

Данные, используемые в примерах:



Отключите сортировку поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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:**

р *MinString* — функция диаграммы (страница 360)

### RangeMissingCount

**RangeMissingCount()** возвращает количество нечисловых значений (включая NULL) в выражении или поле.

**Синтаксис:**

```
RangeMissingCount (first_expr[, Expression])
```

**Возвращаемые типы данных:** целое число

**Аргументы:**

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.



## 5 Функции в скриптах и выражениях диаграммы

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	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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Возвращает 2, поскольку над этой строкой нет строк, поэтому 2 из трех 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 (страница 202)*

### RangeMode

**RangeMode()** находит наиболее часто встречающееся значение (значение режима) в выражении или поле.

**Синтаксис:**

```
RangeMode (first_expr {, Expression})
```

**Возвращаемые типы данных:** число

**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.

**Ограничения:**

Если одинаково часто встречаются несколько значений, возвращается значение NULL.

**Примеры и результаты:**

Примеры	Результаты
RangeMode (1,2,9,2,4)	Возвращает 2
RangeMode ('a',4,'a',4)	Возвращает NULL

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты														
<code>RangeMode (null( ))</code>	Возвращает NULL														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <p>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 ];</p>	<p>Результирующая таблица показывает возвращенные значения функции <b>MyRangeMode</b> для каждой записи в таблице.</p> <table><thead><tr><th>RangeID</th><th>MyRangMode</th></tr></thead><tbody><tr><td>1</td><td>-</td></tr><tr><td>2</td><td>-</td></tr><tr><td>3</td><td>8</td></tr><tr><td>4</td><td>-</td></tr><tr><td>5</td><td>5</td></tr><tr><td>6</td><td>-</td></tr></tbody></table>	RangeID	MyRangMode	1	-	2	-	3	8	4	-	5	5	6	-
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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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:**

р *Mode* — функция диаграммы (страница 188)

### 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* (страница 656).

**Синтаксис:**

```
RangeNPV (discount_rate, value[,value][, Expression])
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
discount_rate	Процентная ставка за период.
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.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты														
<code>RangeNPV(0.1,-10000,3000,4200,6800)</code>	Возвращает 1188,44														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ' ');</pre>	<p>Результирующая таблица показывает возвращенные значения функции RangeNPV для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>RangeNPV</th></tr><tr><td>1</td><td>\$-49.13</td></tr><tr><td>2</td><td>\$777.78</td></tr><tr><td>3</td><td>\$98.77</td></tr><tr><td>4</td><td>\$25.51</td></tr><tr><td>5</td><td>\$250.83</td></tr><tr><td>6</td><td>\$20.40</td></tr></table>	RangeID	RangeNPV	1	\$-49.13	2	\$777.78	3	\$98.77	4	\$25.51	5	\$250.83	6	\$20.40
RangeID	RangeNPV														
1	\$-49.13														
2	\$777.78														
3	\$98.77														
4	\$25.51														
5	\$250.83														
6	\$20.40														

**См. также:**

р *Функции между записями (страница 569)*

### RangeNullCount

**RangeNullCount()** находит значения NULL в выражении или поле.

**Синтаксис:**

```
RangeNullCount (first_expr [, Expression])
```

**Возвращаемые типы данных:** целое число

**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
<code>RangeNullCount (1,2,4)</code>	Возвращает 0
<code>RangeNullCount (5, 'abc')</code>	Возвращает 0
<code>RangeNullCount (null( ), null( ))</code>	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**.

### 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:

р *NullCount* — функция диаграммы (страница 205)

## RangeNumericCount

**RangeNumericCount()** находит числовые значения в выражении или поле.

### Синтаксис:

```
RangeNumericCount (first_expr[, Expression])
```

**Возвращаемые типы данных:** целое число

### 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
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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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 (страница 208)*

### 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:** dual

**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 (страница 190)

### RangeSkew

**RangeSkew()** возвращает значение, соответствующее асимметрии диапазона чисел.

**Синтаксис:**

```
RangeSkew (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

**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.

**Ограничения:**

Если числовые значения не найдены, возвращается значение NULL.

**Примеры и результаты:**

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:

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 (страница 270)

### RangeStdev

**RangeStdev()** находит стандартное отклонение диапазона чисел.

**Синтаксис:**

```
RangeStdev(first_expr[, Expression])
```

**Возвращаемые типы данных:** число

**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.

**Ограничения:**

Если числовые значения не найдены, возвращается значение NULL.

### Примеры и результаты:

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:

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 (страница 273)*

### RangeSum

**RangeSum()** возвращает сумму диапазона значений. Все нечисловые значения приравниваются к 0 в отличие от оператора +.

#### Синтаксис:

```
RangeSum (first_expr[, Expression])
```

**Возвращаемые типы данных:** число

#### 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.

#### Ограничения:

Функция **RangeSum** считает все нечисловые значения равными 0, в отличие от оператора +.

#### Примеры и результаты:

Примеры	Результаты
RangeSum (1,2,4)	Возвращает 7
RangeSum (5, 'abc')	Возвращает 5
RangeSum (null( ))	Возвращает 0

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ];</pre>	<p>Результирующая таблица показывает возвращенные значения функции MyRangeSum для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>MyRangeSum</th></tr><tr><td>1</td><td>21</td></tr><tr><td>2</td><td>12</td></tr><tr><td>3</td><td>18</td></tr><tr><td>4</td><td>38</td></tr><tr><td>5</td><td>19</td></tr><tr><td>6</td><td>15</td></tr></table>	RangeID	MyRangeSum	1	21	2	12	3	18	4	38	5	19	6	15
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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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:

р *Sum* — функция диаграммы (страница 193)  
р *Above* — функция диаграммы (страница 573)

## RangeTextCount

**RangeTextCount()** возвращает текстовые значения в выражении или поле.

### Синтаксис:

```
RangeTextCount (first_expr[, Expression])
```

**Возвращаемые типы данных:** целое число

### 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
RangeTextCount (1,2,4)	Возвращает 0
RangeTextCount (5,'abc')	Возвращает 1
RangeTextCount (null( ))	Возвращает 0

Пример с выражением:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

## 5 Функции в скриптах и выражениях диаграммы

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 поля **MyField**, чтобы убедиться, что пример работает, как ожидается.

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:**

р *TextCount* — функция диаграммы (страница 211)

### 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:** число

**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
<code>RangeXIRR(-2500, '2008-01-01', 2750, '2008-09-01')</code>	Возвращает 0,1532

**См. также:**

р *RangeIRR* (страница 631)

### 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:** число

**Аргументы:**

Аргумент	Описание
discount_rate	Процентная ставка за период.



## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
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.

Примеры	Результаты														
<code>RangeXNPV(0.1, -2500, '2008-01-01', 2750, '2008-09-01')</code>	Возвращает 80,25														
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ' ');</pre>	<p>Результирующая таблица показывает возвращенные значения функции RangeXNPV для каждой записи в таблице.</p> <table><tr><th>RangeID</th><th>RangeXNPV</th></tr><tr><td>1</td><td>\$-49.13</td></tr><tr><td>2</td><td>\$777.78</td></tr><tr><td>3</td><td>\$98.77</td></tr><tr><td>4</td><td>\$25.51</td></tr><tr><td>5</td><td>\$250.83</td></tr><tr><td>6</td><td>\$20.40</td></tr></table>	RangeID	RangeXNPV	1	\$-49.13	2	\$777.78	3	\$98.77	4	\$25.51	5	\$250.83	6	\$20.40
RangeID	RangeXNPV														
1	\$-49.13														
2	\$777.78														
3	\$98.77														
4	\$25.51														
5	\$250.83														
6	\$20.40														

## 5.20 Функции ранжирования в диаграммах

Эти функции могут использоваться только в выражениях диаграмм.



*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]])
```

## 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.

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:** dual

### 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 префикс <b>TOTAL</b> , 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.

## 5 Функции в скриптах и выражениях диаграммы

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:

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:

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.

## 5 Функции в скриптах и выражениях диаграммы

Примеры	Результаты
Пример 1. Создайте таблицу с измерениями Customer и Sales и мерой Rank(Sales)	<p>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.</p> <p>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.</p>
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 <b>mode</b> и <b>fmt</b> установлены значения 1 и 2 соответственно.

Результаты для примера 1 — таблица отсортирована по значению Customer:

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5

## 5 Функции в скриптах и выражениях диаграммы

Customer	Sales	Rank(Sales)
Astrida	60	4
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

Результаты для примера 1 — таблица отсортирована по значению Sales:

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 (страница 193)

### 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: dual



Эта функция доступна только при работе со сводными таблицами. Во всех других типах диаграмм она возвращает значение NULL.

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 префикс <b>TOTAL</b> , 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.

## 5 Функции в скриптах и выражениях диаграммы

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:

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.
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:

Значение	Описание
0 (по умолчанию)	Низкое значение <b>&amp;</b> - <b>'&amp;'</b> 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 )
```

### 5.21 Функции статистического распределения

Все описанные ниже функции статистического распределения реализованы в программе Qlik Sense с помощью библиотеки функций Cephес. Ссылки и подробная информация об используемых алгоритмах, точности и т. д. см. на веб-странице <http://www.netlib.org/cephes/>. Библиотека функций Cephес используется с разрешения.

Функции статистического распределения DIST измеряют вероятность функции распределения в точке распределения, заданной предоставленным значением. Функции INV вычисляют значение, заданное вероятностью распределения. В отличие от них, группы функций статистического агрегирования вычисляют агрегированные значения последовательностей статистических тестовых значений для проверки различных статистических гипотез.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

#### Обзор функций статистического распределения

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

##### CHIDIST

**CHIDIST()** возвращает одностороннюю вероятность распределения  $\chi^2$ . Распределение  $\chi^2$  ассоциируется с критерием  $\chi^2$ .

```
CHIDIST (value, degrees_freedom)
```

##### CHIINV

**CHIINV()** возвращает обратную одностороннюю вероятность распределения  $\chi^2$ .

```
CHIINV (prob, degrees_freedom)
```

##### NORMDIST

**NORMDIST()** возвращает накопленное нормальное распределение указанного среднего значения и стандартного отклонения. Если значение mean = 0, а значение standard\_dev = 1, функция возвращает стандартное нормальное распределение.

```
NORMDIST (value, mean, standard_dev)
```

##### NORMINV

**NORMINV()** возвращает противоположное нормальное распределение указанного среднего значения и стандартного отклонения.

```
NORMINV (prob, mean, standard_dev)
```

##### TDIST

**TDIST()** возвращает вероятность t-распределения, в котором числовое значение является вычисляемым значением t, для которого должна подсчитываться вероятность.



## 5 Функции в скриптах и выражениях диаграммы

**TDIST** (value, degrees\_freedom, tails)

### TINV

**TINV()** возвращает t-значение t-распределения в виде функции вероятности и степеней свободы.

**TINV** (prob, degrees\_freedom)

### FDIST

**FDIST()** возвращает вероятность F-распределения.

**FDIST** (value, degrees\_freedom1, degrees\_freedom2)

### FINV

**FINV()** возвращает обратную вероятность F-распределения.

**FINV** (prob, degrees\_freedom1, degrees\_freedom2)

См. также:

р *Функции статистического агрегирования (страница 225)*

## CHIDIST

**CHIDIST()** возвращает одностороннюю вероятность распределения  $\chi^2$ . Распределение  $\chi^2$  ассоциируется с критерием  $\chi^2$ .

**Синтаксис:**

**CHIDIST**(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.

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.

## 5 Функции в скриптах и выражениях диаграммы

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:**

Аргумент	Описание
prob	Вероятность, связанная с распределением $\chi^2$ . Значение должно быть числом от 0 до 1.
degrees_freedom	Целое число, которое указывает число степеней свободы.

Эта функция связана с функцией **CHIDIST** следующим образом:

If prob = CHIDIST(value,df), then CHIINV(prob, df) = value.

**Ограничения:**

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
CHIINV(0.9237827, 15 )	Возвращает 8,0000

### FDIST

**FDIST()** возвращает вероятность F-распределения.

**Синтаксис:**

```
FDIST(value, degrees_freedom1, degrees_freedom2)
```

**Возвращаемые типы данных:** число

**Аргументы:**

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение. Элемент <b>Value</b> не должен быть отрицательным.
degrees_freedom1	Положительное целое число, которое указывает число степеней свободы числителя.
degrees_freedom2	Положительное целое число, которое указывает число степеней свободы знаменателя.

Эта функция связана с функцией **FINV** следующим образом:

If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value.

**Ограничения:**

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
FDIST(15, 8, 6)	Возвращает 0,0019

## FINV

**FINV()** возвращает обратную вероятность F-распределения.

**Синтаксис:**

```
FINV(prob, degrees_freedom1, degrees_freedom2)
```

**Возвращаемые типы данных:** число

**Аргументы:**

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
<code>FINV( 0.0019369, 8, 6)</code>	Возвращает 15,0000

## NORMDIST

**NORMDIST()** возвращает накопленное нормальное распределение указанного среднего значения и стандартного отклонения. Если значение `mean` = 0, а значение `standard_dev` = 1, функция возвращает стандартное нормальное распределение.

### Синтаксис:

```
NORMDIST(value, mean, standard_dev)
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
<code>value</code>	Значение, при котором необходимо оценить распределение.
<code>mean</code>	Значение, указывающее среднее арифметическое для распределения.
<code>standard_dev</code>	Положительное значение, указывающее стандартное отклонение распределения.

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
<code>NORMDIST( 0.5, 0, 1)</code>	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:

Аргумент	Описание
prob	Вероятность, связанная с нормальным распределением. Значение должно быть числом от 0 до 1.
mean	Значение, указывающее среднее арифметическое для распределения.
standard_dev	Положительное значение, указывающее стандартное отклонение распределения.

Эта функция связана с функцией **NORMDIST** следующим образом:

If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value.

### Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
NORMINV( 0.6914625, 0, 1 )	Возвращает 0,5000

## TDIST

**TDIST()** возвращает вероятность t-распределения, в котором числовое значение является вычисляемым значением t, для которого должна подсчитываться вероятность.

### Синтаксис:

```
TDIST(value, degrees_freedom, tails)
```

**Возвращаемые типы данных:** число

### Аргументы:

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение и которое не должно быть отрицательным.

## 5 Функции в скриптах и выражениях диаграммы

Аргумент	Описание
degrees_freedom	Положительное целое число, которое указывает число степеней свободы.
tails	Должно составлять 1 (одностороннее распределение) или 2 (двустороннее распределение).

Эта функция связана с функцией **TINV** следующим образом:

If prob = TDIST(value, df ,2), then TINV(prob, df) = value.

### Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
TDIST(1, 30, 2)	Возвращает 0,3253

## TINV

**TINV()** возвращает t-значение t-распределения в виде функции вероятности и степеней свободы.

### Синтаксис:

```
TINV(prob, degrees_freedom)
```

**Возвращаемые типы данных:** число

### Аргументы:

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

### 5.22 Строковые функции

В этом разделе описаны функции для обработки и управления строками. В приведенных ниже функциях параметры — это выражения, в которых элемент **s** должен интерпретироваться как строка.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм, кроме функции **Evaluate**, которую можно использовать только в скрипте загрузки данных.

#### Обзор строковых функций

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

##### Capitalize

**Capitalize()** возвращает строку со всеми словами, которые начинаются с заглавной буквы.

```
Capitalize (text)
```

##### Chr

**Chr()** возвращает символ Unicode, соответствующий входному целому числу.

```
Chr (int)
```

##### Evaluate

**Evaluate()** определяет, можно ли входную текстовую строку рассматривать в качестве допустимого выражения Qlik Sense. Если да, то возвращает значение выражения в качестве строки. Если входная строка не является допустимым выражением, будет возвращено значение NULL.

```
Evaluate (expression_text)
```

##### FindOneOf

**FindOneOf()** выполняет поиск в строке, чтобы найти положение вхождения любого символа из набора указанных символов. Положение вхождения любого символа из набора для поиска возвращается, если указан третий аргумент (значение больше 1). Если совпадений не найдено, возвращается значение 0.

```
FindOneOf (text, char_set[, count])
```

##### Hash128

**Hash128()** возвращает 128-разрядный хэш сочетания значений входного выражения. Результат — строка из 22 символов.

```
Hash128 (expr[, expression])
```

### Hash160

**Hash160()** возвращает 160-разрядный хэш сочетания значений входного выражения. Результат — строка из 27 символов.

```
Hash160 (expr{, expression})
```

### Hash256

**Hash256()** возвращает 256-разрядный хэш сочетания значений входного выражения. Результат — строка из 43 символов.

```
Hash256 (expr{, expression})
```

### Index

**Index()** выполняет поиск в строке, чтобы найти n-ное положение вхождения указанной подстроки. Дополнительный третий аргумент определяет значение n (1, если игнорируется). Если указано отрицательное значение, поиск выполняется с конца строки. Позиции в строке нумеруются от 1 и далее.

```
Index (text, substring[, count])
```

### KeepChar

**KeepChar()** возвращает строку, состоящую из первой строки «text», за вычетом всех символов, НЕ содержащихся во второй строке «keep\_chars».

```
KeepChar (text, keep_chars)
```

### Left

**Left()** возвращает строку, состоящую из первых (крайних слева) символов входной строки, где число символов определяется вторым аргументом.

```
Left (text, count)
```

### Len

**Len()** возвращает длину входной строки.

```
Len (text)
```

### 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



---

## 5 Функции в скриптах и выражениях диаграммы

---

аргументом «start», и возвращает количество символов, определенных третьим аргументом «count». Если «count» отсутствует, возвращается оставшая 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 (char )
```

### **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.

## 5 Функции в скриптах и выражениях диаграммы

```
SubStringCount ( text, substring)
```

### TextBetween

**TextBetween()** возвращает текст входной строки, заключенный между символами, указанными в качестве разделителей.

```
TextBetween (text, sub_string)
```

### 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

Примеры и результаты:

Пример	Результат
Capitalize ( 'my little pony' )	Возвращает 'My Little Pony'
Capitalize ( 'AA bb CC Dd' )	Возвращает 'Aa Bb Cc Dc'

## 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'

### 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:** dual



*Эта строковая функция не может использоваться в выражениях диаграмм.*

Примеры и результаты:

Пример	Результат
Evaluate ( 5 * 8 )	Возвращает '40'

### 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 число

**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.

Примеры и результаты:

Пример	Результат
<code>FindOneOf( 'my example text string', 'et%s')</code>	Возвращает «4».
<code>FindOneOf( 'my example text string', 'et%s', 3)</code>	Возвращает «12». Потому что поиск выполняется для любого из символов: e, t, % или s, «t» является третьим элементом и его позиция 12.
<code>FindOneOf( 'my example text string', 'x%&amp;')</code>	Возвращает «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:**

```
hash128 ( 'abc', 'xyz', '123' )  
hash128 ( Region, Year, Month )
```

### Hash160

**Hash160()** возвращает 160-разрядный хэш сочетания значений входного выражения. Результат — строка из 27 символов.

**Синтаксис:**

```
Hash160 (expr{, expression})
```

**Возвращаемые типы данных:** строка

**Пример:**

```
hash160 ( 'abc', 'xyz', '123' )  
hash160 ( Region, Year, Month )
```

### Hash256

**Hash256()** возвращает 256-разрядный хэш сочетания значений входного выражения. Результат — строка из 43 символов.

**Синтаксис:**

```
Hash256 (expr{, expression})
```

**Возвращаемые типы данных:** строка

**Пример:**

```
hash256 ( 'abc', 'xyz', '123' )  
hash256 ( Region, Year, Month )
```

### 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 число

**Аргументы:**

Аргумент	Описание
text	Оригинальная строка.
substring	Строка символов для поиска в text.
count	Определяет, какое вхождение символов <b>substring</b> 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 <b>Date</b> = 1997-07-14	Возвращает 1997
Mid( Date, Index( Date, '-' , 2 ) -2, 2 ) where <b>Date</b> = 1997-07-14	Возвращает 07

### 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:

Аргумент	Описание
text	Оригинальная строка.
keep_chars	Строка, содержащая символы в text, которую необходимо сохранить.

### Примеры и результаты:

Пример	Результат
KeepChar ( 'a1b2c3', '123' )	Возвращает «123».
KeepChar ( 'a1b2c3', '1234' )	Возвращает «123».
KeepChar ( 'a1b22c3', '1234' )	Возвращает «1223».
KeepChar ( 'a1b2c3', '312' )	Возвращает «123».

### См. также:

р *PurgeChar* (страница 681)

## 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 <b>text</b> .

---

## 5 Функции в скриптах и выражениях диаграммы

---

Примеры и результаты:

Пример	Результат
<code>Left('abcdef', 3)</code>	Возвращает 'abc'

См.: Функция *Index* (страница 677), которая обеспечивает более сложный анализ строк.

### Len

**Len()** возвращает длину входной строки.

**Синтаксис:**

**Len** (text)

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
<code>Len('Peter')</code>	Возвращает '5'

### Lower

**Lower()** преобразует все символы входной строки в нижний регистр.

**Синтаксис:**

**Lower** (text)

**Возвращаемые типы данных:** строка

Примеры и результаты:

Пример	Результат
<code>Lower('abcD')</code>	Возвращает 'abcd'

### LTrim

**LTrim()** возвращает входную строку без начальных пробелов.

**Синтаксис:**

**LTrim** (text)

**Возвращаемые типы данных:** строка

Примеры и результаты:

Пример	Результат
<code>LTrim( ' abc' )</code>	Возвращает 'abc'
<code>LTrim( 'abc ' )</code>	Возвращает 'abc '

**См. также:**

р *RTrim* (страница 683)

### 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 оставшая часть входной строки. Первый символ во входной строке имеет номер 1.

**Синтаксис:**

```
Mid(text, start[, count])
```

**Возвращаемые типы данных:** строка

**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 <b>start</b> .

Примеры и результаты:

Пример	Результат
<code>mid('abcdef', 3 )</code>	Возвращает «cdef»
<code>mid('abcdef', 3, 2 )</code>	Возвращает «cd»

**См. также:**

р *Index* (страница 677)



### Ord

**Ord()** возвращает номер кодовой точки Unicode первого символа входной строки.

**Синтаксис:**

```
Ord(char)
```

**Возвращаемые типы данных:** целое число

Примеры и результаты:

Пример	Результат
ord('A')	Возвращает целое число 65.
ord('Ab')	Возвращает целое число 65.

### 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:**

Аргумент	Описание
text	Оригинальная строка.
remove_chars	Строка, содержащая символы в text, которую необходимо удалить.

**Возвращаемые типы данных:** строка

Примеры и результаты:

Пример	Результат
PurgeChar ( 'a1b2c3','123' )	Возвращает «abc»
PurgeChar ( 'a1b2c3','312' )	Возвращает «abc»

**См. также:**

р *KeepChar* (страница 677)

### 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:

Аргумент	Описание
text	Оригинальная строка.
repeat_count	Определяет, сколько раз символы в строке <b>text</b> повторяются в выводимой строке.

Примеры и результаты:

Пример	Результат
Repeat( ' * ', rating ) when <b>rating</b> = 4	Возвращает '****'

### Replace

**Replace()** возвращает строку после замены всех вхождений определенной подстроки во входной строке на другую подстроку. Функция не рекурсивная и работает слева направо.

#### Синтаксис:

```
Replace (text, from_str, to_str)
```

**Возвращаемые типы данных:** строка

#### Аргументы:

Аргумент	Описание
text	Оригинальная строка.
from_str	Строка, встречающаяся один или несколько раз во входной строке <b>text</b> .
to_str	Строка, заменяющая все вхождения <b>from_str</b> в строке <b>text</b> .

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:**

Аргумент	Описание
text	Оригинальная строка.
count	Определяет количество символов, которые необходимо включить из правой части строки <b>text</b> .

Примеры и результаты:

Пример	Результат
Right('abcdef', 3)	Возвращает 'def'

### RTrim

**RTrim()** возвращает входную строку без конечных пробелов.

**Синтаксис:**

```
RTrim(text)
```

**Возвращаемые типы данных:** строка

Примеры и результаты:

Пример	Результат
RTrim( ' abc' )	Возвращает 'abc'
RTrim( 'abc ' )	Возвращает 'abc'

См. также:

р *LTrim* (страница 679)

### 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:

Аргумент	Описание
text	Оригинальная строка. Это может быть неизменяемый текст, переменная, расширение со знаком доллара или другое расширение.
delimiter	Символ во входной строке <b>text</b> , разделяющий строку на части.
field_no	Дополнительный третий аргумент, являющийся целым числом, который указывает, какие подстроки родительской строки <b>text</b> необходимо вернуть. Используйте значение 1 для возврата первой подстроки, значение 2 для возврата второй подстроки и так далее. Отрицательное значение означает, что подстрока извлекается из правой части строки. Таким образом, поиск по строке осуществляется справа налево, а не слева направо, как бывает, если параметр <b>field_no</b> имеет положительное значение.



Функцию **SubField()** можно использовать вместо сложных комбинаций таких функций, как **Len()**, **Right()**, **Left()**, **Mid()** и другие строковые функции.

#### Примеры и результаты:

Пример	Результат
<code>SubField(S, ';' ,2)</code>	Возвращает 'cde', если <b>S</b> = 'abc;cde;efg'.

## 5 Функции в скриптах и выражениях диаграммы

Пример	Результат															
SubField(S, ';' ,1)	Возвращает NULL, если элемент <b>S</b> — пустая строка.															
SubField(S, ';' ,1)	Возвращает пустую строку, если элемент <b>S</b> =';'.															
<p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>FullName: LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames: Load Name, SubField(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table FullName;</pre>	<table><tr><th>Name</th><th>FirstName</th><th>Surname</th></tr><tr><td>Dave</td><td>Dave</td><td>Owen</td></tr><tr><td>Owen</td><td></td><td></td></tr><tr><td>Joe</td><td>Joe</td><td>Tem</td></tr><tr><td>Tem</td><td></td><td></td></tr></table>	Name	FirstName	Surname	Dave	Dave	Owen	Owen			Joe	Joe	Tem	Tem		
Name	FirstName	Surname														
Dave	Dave	Owen														
Owen																
Joe	Joe	Tem														
Tem																
<p>Предполагается, что уже есть переменная, которая содержит имя путиvMyPath,</p> <pre>Set vMyPath=\\Users\\ext_jrb\\Documents\\Qlik\\Sense\\Apps;</pre>	<p>В диаграмме текста и изображения можно добавить меру, такую как:</p> <p>SubField(vMyPath, '\\',-3), что отразится в 'Qlik', потому что это подстрока третья, если считать с правого конца переменной vMyPath.</p>															

Пример	Результат		
<p>В этом примере показано, как с помощью нескольких экземпляров функции <b>Subfield()</b>, в каждом из которых не указан параметр <b>field_no</b>, из одного оператора <b>LOAD</b> создать декартово произведение всех возможных комбинаций. Параметр <b>DISTINCT</b> используется, чтобы не допустить создания дубликата записи.</p> <p>Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.</p> <pre>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 ' ');</pre>	<b>Instrument</b>	<b>Player</b>	<b>Project</b>
	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

### 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 число

#### Аргументы:

Аргумент	Описание
text	Оригинальная строка.
sub_string	Строка, встречающаяся один или несколько раз во входной строке <b>text</b> .

Примеры и результаты:

Пример	Результат
<code>substringCount ( 'abcdefgcdxyz', 'cd' )</code>	Возвращает '2'
<code>substringCount ( 'abcdefgcdxyz', 'dc' )</code>	Возвращает '0'

### 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 <b>text</b> .
delimiter2	Указывает второй символ-разделитель (или строку) для поиска в <b>text</b> .
n	Указывает, между каким вхождением пары разделителей выполнять поиск. Например, значение 2 возвращает символы между вторым вхождением разделителя1 и вторым вхождением разделителя2.

Примеры и результаты:

Пример	Результат
<code>TextBetween('&lt;abc&gt;', '&lt;', '&gt;')</code>	Возвращает 'abc'
<code>TextBetween('&lt;abc&gt;&lt;de&gt;', '&lt;', '&gt;', 2)</code>	Возвращает 'de'

### Trim

**Trim()** возвращает входную строку без начальных и конечных пробелов.

**Синтаксис:**

```
Trim(text)
```

**Возвращаемые типы данных:** строка

Примеры и результаты:

Пример	Результат
<code>Trim( ' abc' )</code>	Возвращает 'abc'
<code>Trim( 'abc ' )</code>	Возвращает 'abc'
<code>Trim( ' abc ' )</code>	Возвращает 'abc'

### 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

Примеры и результаты:

Пример	Результат
<code>upper(' abcd')</code>	Возвращает 'ABCD'

## 5.23 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()



## 5 Функции в скриптах и выражениях диаграммы

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.

```
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 ( )
```

### GetCollationLocale

Эта функция скрипта возвращает имя культуры используемой локали сортировки. Если переменная CollationLocale не задана, возвращается локаль машины фактического пользователя.

```
GetCollationLocale ( )
```

### GetObjectField

Эта функция возвращает имя измерения. **Index** — дополнительное целое число, обозначающее используемое измерение, которое необходимо вернуть.

```
GetObjectField — функция диаграммы([index])
```

### GetRegistryString

## 5 Функции в скриптах и выражениях диаграммы

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 программе Qlik Sense Desktop эта функция всегда возвращает «Personal\Me».*

### ProductVersion

Эта функция возвращает полную версию программы Qlik Sense и номер сборки в виде строки.

**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()

**См. также:**

р [GetFolderPath](#) (страница 517)

### GetObjectField — функция диаграммы

Эта функция возвращает имя измерения. **Index** — дополнительное целое число, обозначающее используемое измерение, которое необходимо вернуть.



*В метке заголовка диаграммы эту функцию использовать невозможно.*

#### Синтаксис:

```
GetObjectField ([index])
```

#### Пример:

```
GetObjectField(2)
```

### 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.

#### 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 ()
```



*Другие состояния можно определить и назначить только с помощью Qlik Engine API.*

#### Пример 1:

```
Динамический текст  
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

### Пример 2:

```
Динамические цвета
if(StateName() = 'Group 1', rgb(152, 171, 206),
    if(StateName() = 'Group 2', rgb(187, 200, 179),
        rgb(210, 210, 210)
    )
)
```

## 5.24 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($(t))) as Field,  
FieldNumber(FieldNumber($(f),TableName($(t))),TableName($(t))) as FieldNo  
Autogenerate 1;  
Next f  
Next t;
```

Результирующая таблица Tables будет выглядеть так:

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:

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:

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:

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:

Аргумент	Описание
table_name	Имя таблицы.

### Пример:

```
LET a = NoOfRows('tab1');
```

## 5.25 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  $-1 \leq x \leq 1$ . Результат находится в интервале от 0 до  $\pi$ .

```
acos ( x )
```

### **sin**

Синус **x**. Результат находится в интервале от -1 до +1.

```
sin ( x )
```

### **asin**

Арсинус **x**. Функция определяется, только если  $-1 \leq x \leq 1$ . Результат находится в интервале от  $-\pi/2$  до  $\pi/2$ .

```
asin ( x )
```

### **tan**

Тангенс **x**. Результат — действительное число.

```
tan ( x )
```

### **atan**

Арктангенс **x**. Результат находится в интервале от  $-\pi/2$  до  $\pi/2$ .

```
atan ( x )
```

### **atan2**

Двухмерное представление функции арктангенса. Возвращает угол между началом координат и



---

## 5 Функции в скриптах и выражениях диаграммы

---

точкой с координатами **x** и **y**. Результат находится в интервале от  $-\pi$  до  $+\pi$ .

```
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 )
```

### 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 абсолютные или относительные пути в скрипте загрузки данных или функциях и переменных, предоставляющих доступ к файловой системе.

Однако, поскольку абсолютные и относительные пути поддерживаются 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.*



*Невозможно отключить стандартный режим в Qlik Sense Cloud. Другие режимы не поддерживаются.*

## 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

Переменная	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
Floppy	Не поддерживается	Поддерживается	Возвращает буквенное обозначение первого найденного дисководов 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.

## 6 Ограничение доступа к файловой системе

Переменная	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
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.
\$(include=...)	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Переменная <b>Include/Must_Include</b> указывает файл, содержащий текст, который необходимо включить в скрипт и который рассматривается в качестве кода скрипта. Можно сохранять часть кода скрипта в отдельный текстовый файл и использовать его в разных приложениях. Эта переменная определяется пользователем.

### Обычные операторы скриптов

Оператор	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
Binary	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Оператор <b>binary</b> используется для загрузки данных из другого приложения.
Connect	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Оператор <b>CONNECT</b> 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: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Оператор <b>Directory</b> задает каталог, в котором будет выполняться поиск файлов данных в последующих операторах <b>LOAD</b> до создания нового оператора <b>Directory</b> .

## 6 Ограничение доступа к файловой системе

Оператор	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
Execute	Не поддерживается	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Оператор <b>Execute</b> используется для запуска других программ в ходе загрузки данных Qlik Sense. Например, для выполнения необходимых преобразований.
LOAD from ...	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Возвращает строку обзора в выполняемый модуль Qlik Sense.
Store into ...	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Возвращает корневой каталог выполняемого модуля Qlik Sense.

### Операторы управления скриптом

Оператор	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
For each... filelist mask/dirlist mask	Поддерживаемый ввод: подключение к библиотеке  Выходные данные: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь  Выходные данные: подключение к библиотеке или абсолютный путь, в зависимости от ввода	Синтаксис filelist mask создает разделенный запятыми список всех файлов в текущем каталоге, соответствующих маске имени файла <b>filelist mask</b> . Синтаксис dirlist mask создает разделенный запятыми список всех каталогов в текущем каталоге, соответствующих маске имени каталога.

### Функции файлов

Функция	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
Attribute()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Возвращает значение метатегов 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.

## 6 Ограничение доступа к файловой системе

Функция	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
FileDir()	Выходные данные: подключение к библиотеке	Выходные данные: подключение к библиотеке или абсолютный путь, в зависимости от ввода	Функция <b>FileDir</b> возвращает строку, содержащую путь к каталогу табличного файла, читаемого в текущий момент.
FilePath()	Выходные данные: подключение к библиотеке	Выходные данные: подключение к библиотеке или абсолютный путь, в зависимости от ввода	Функция <b>FilePath</b> возвращает строку, содержащую полный путь табличного файла, читаемого в текущий момент.
FileSize()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Функция <b>FileSize</b> возвращает целое, содержащее размер в байтах файла filename, или, если не указан файл filename, табличного файла, читаемого в текущий момент.



## 6 Ограничение доступа к файловой системе

Функция	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
FileTime()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Функция <b>FileTime</b> возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.
GetFolderPath()	Не поддерживается	Выходные данные: абсолютный путь	Функция <b>GetFolderPath</b> возвращает значение функции Microsoft Windows <i>SHGetFolderPath</i> . Данная функция берет в качестве значения ввода имя папки Microsoft Windows и возвращает полный путь папки.

## 6 Ограничение доступа к файловой системе

Функция	Стандартный режим/Qlik Sense Cloud	Устаревший режим	Определение
QvdCreateTime()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.
QvdFieldName()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Эта функция скрипта возвращает имя числа поля <b>fieldno</b> , если оно существует в файле QVD, в противном случае — значение NULL.
QvdNoOfFields()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Эта функция скрипта возвращает число полей в файле QVD.
QvdNoOfRecords()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Эта функция скрипта возвращает число записей, находящихся в настоящее время в файле QVD.
QvdTableName()	Поддерживаемый ввод: подключение к библиотеке	Поддерживаемый ввод: подключение к библиотеке или абсолютный/относительный путь	Эта функция скрипта возвращает имя таблицы, хранящейся в файле QVD.

### Системные функции

Функция	Стандартный режим/Qlik Sense Cloud	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.*



*Невозможно отключить стандартный режим в Qlik Sense Cloud.*

### 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*.

Выполните следующие действия.

1. Откройте файл `C:\Users\{user}\Documents\Qlik\Sense\Settings.ini` в текстовом редакторе.
2. Измените `StandardReload=1` на `StandardReload=0`.
3. Сохраните файл и запустите программу Qlik Sense Desktop, которая откроется в устаревшем режиме.

Для `StandardReload` доступны следующие параметры:

- 1 (стандартный режим)
- 0 (устаревший режим)

## 7 Функции и операторы QlikView, не поддерживаемые в Qlik Sense

Большинство функций и операторов, которые можно использовать в скриптах загрузки QlikView и выражениях диаграмм, также поддерживаются в Qlik Sense, но есть несколько исключений.

### 7.1 Операторы скрипта, не поддерживаемые в Qlik Sense

В этом списке представлены операторы скрипта QlikView, которые не поддерживаются в Qlik Sense.

Оператор	Комментарии
Command	Используйте <b>SQL</b> .
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. 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 этом списке представлены операторы скрипта QlikView, которые не рекомендуются для использования в Qlik Sense.

Оператор	Рекомендация
<b>Command</b>	Используйте <b>SQL</b> .
<b>CustomConnect</b>	Используйте <b>Custom Connect</b> .

## 8.2 Параметры оператора скрипта, не рекомендуемые в Qlik Sense

В этом списке представлены параметры оператора скрипта QlikView, которые не рекомендуются для использования в Qlik Sense.

Оператор	Параметры
<b>Buffer</b>	Используйте <b>Incremental</b> вместо: <ul style="list-style-type: none"><li>• <b>Inc</b> (не рекомендуется)</li><li>• <b>Incr</b> (не рекомендуется)</li></ul>

## 8 Функции и операторы, не рекомендуемые в Qlik Sense

---

Оператор	Параметры
<b>LOAD</b>	<p>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:</p> <ul style="list-style-type: none"><li>• <b>Bottom</b></li><li>• <b>Cellvalue</b></li><li>• <b>Col</b></li><li>• <b>Colmatch</b></li><li>• <b>Colsplit</b></li><li>• <b>Colxtr</b></li><li>• <b>Compound</b></li><li>• <b>Contain</b></li><li>• <b>Equal</b></li><li>• <b>Every</b></li><li>• <b>Expand</b></li><li>• <b>Filters</b></li><li>• <b>Intarray</b></li><li>• <b>Interpret</b></li><li>• <b>Length</b></li><li>• <b>Longer</b></li><li>• <b>Numerical</b></li><li>• <b>Pos</b></li><li>• <b>Remove</b></li><li>• <b>Rotate</b></li><li>• <b>Row</b></li><li>• <b>Rowcnd</b></li><li>• <b>Shorter</b></li><li>• <b>Start</b></li><li>• <b>Strcnd</b></li><li>• <b>Top</b></li><li>• <b>Transpose</b></li><li>• <b>Unwrap</b></li><li>• <b>XML: XMLSAX and Pattern is Path</b></li></ul>

---



### 8.3 Функции, не рекомендуемые в Qlik Sense

В этом списке представлены скрипт и функции диаграммы QlikView, которые не рекомендуются для использования в Qlik Sense.

Функция	Рекомендация
<b>NumAvg</b>	Используйте функции над выборкой.
<b>NumCount</b>	См.: <i>Функции над выборкой (страница 619)</i>
<b>NumMax</b>	
<b>NumMin</b>	
<b>NumSum</b>	
<b>QliktechBlue</b>	Используйте другие функции цвета. Для получения этих цветов <b>QliktechBlue()</b> можно заменить <b>RGB(8, 18, 90)</b> , а <b>QliktechGray</b> можно заменить <b>RGB(158, 148, 137)</b> .
<b>QliktechGray</b>	См.: <i>Функции цвета (страница 365)</i>
<b>QlikViewVersion</b>	Используйте <b>ProductVersion</b> . См.: <i>ProductVersion (страница 691)</i>
<b>QVUser</b>	
<b>Year2Date</b>	Используйте <b>YearToDate</b> .
<b>Vrank</b>	Используйте <b>Rank</b> .
<b>WildMatch5</b>	Используйте <b>WildMatch</b> .

#### Префикс **ALL**

В QlikView префикс **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 префикса **ALL** набор выражений не может использоваться, поскольку префикс **ALL** уже определяет набор. По причинам совместимости префикс **ALL** работает в данной версии Qlik Sense, но может быть удален в следующих версиях.