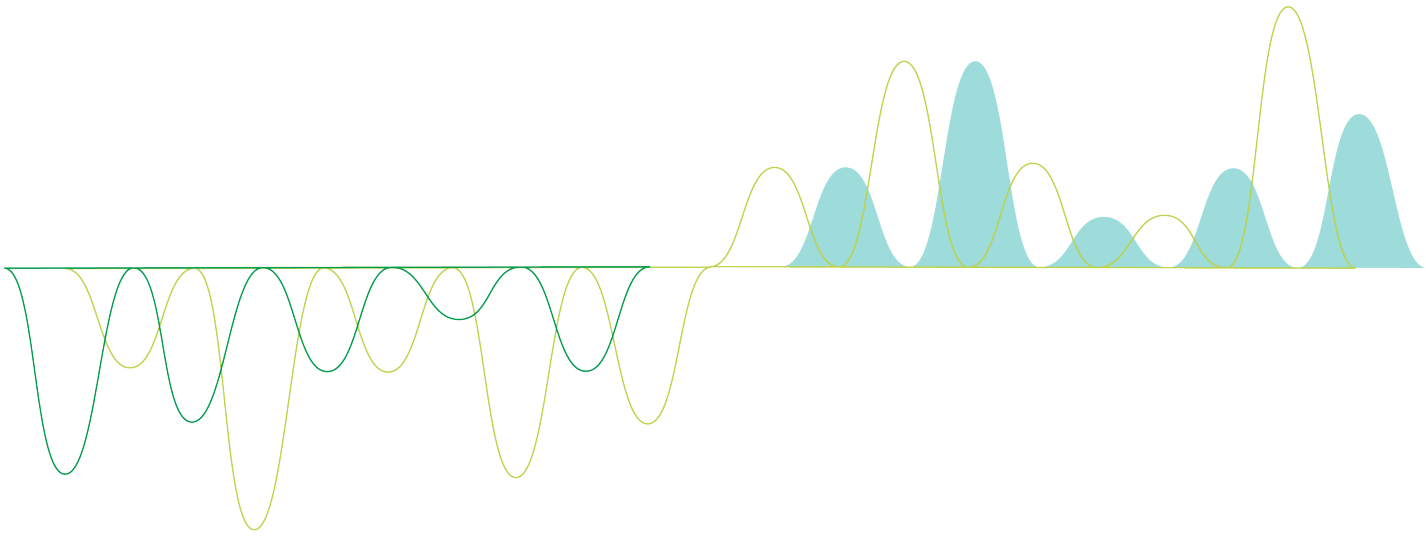
Syntaxe des scripts et fonctions de graphique

Qlik Sense®
November 2022
Copyright © 1993-aaaa} QlikTech International AB. Tous droits réservés.



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

Contents

1 Présentation de Qlik Sense ... 16
1.1 Opportunités offertes par Qlik Sense ... 16
1.2 Fonctionnement de Qlik Sense ... 16
Modèle d'application ... 16
Expérience associative ... 16
Collaboration et mobilité ... 16
1.3 Déploiement de Qlik Sense ... 17
Qlik Sense Desktop ... 17
Qlik Sense Enterprise ... 17
1.4 Procédures d'administration et de gestion d'un site Qlik Sense ... 17
1.5 Extension de Qlik Sense et adaptation à vos propres besoins ... 17
Création d'extensions et d'applications composites ... 17
Création de clients ... 17
Création d'outils serveur ... 17
Connexion à d'autres sources de données ... 18
2 Vue d'ensemble de la syntaxe d'un script ... 19
2.1 Introduction à la syntaxe de script ... 19
2.2 Présentation du code BNF (formalisme Backus-Naur) ... 19
2 Instructions de script et mots-clés ... 21
2.3 Instructions de contrôle de script ... 21
Vue d'ensemble des instructions de contrôle de script ... 21
Call ... 23
Do..loop ... 24
End ... 25
Exit ... 25
Exit script ... 25
For..next ... 26
For each..next ... 27
If..then..elseif..else..end if ... 30
Next ... 31
Sub..end sub ... 31
Switch..case..default..end switch ... 32
To ... 33
2.4 Préfixes de script ... 33
Vue d'ensemble des préfixes de script ... 33
Add ... 38
Buffer ... 39
Concatenate ... 41
Crosstable ... 47
First ... 56
Generic ... 59
Hierarchy ... 65
HierarchyBelongsTo ... 67
Inner ... 69
IntervalMatch ... 70
Join ... 73
Keep ... 83

Left ... 84
Mappage ... 85
Merge ... 86
NoConcatenate ... 91
Only ... 100
Outer ... 100
Chargement partiel ... 101
Replace ... 104
Right ... 106
Sample ... 107
Semantic ... 110
Unless ... 114
When ... 120
2.5 Instructions normales de script ... 126
Vue d'ensemble des instructions normales de script ... 126
Alias ... 133
AutoNumber ... 133
Binary ... 136
Comment field ... 138
Comment table ... 138
Connect ... 139
Declare ... 141
Derive ... 143
Directory ... 144
Disconnect ... 145
Drop ... 146
Drop table ... 147
Execute ... 148
Field/Fields ... 149
FlushLog ... 149
Force ... 150
From ... 151
Load ... 151
Let ... 170
Loosen Table ... 170
Map ... 171
NullAsNull ... 172
NullAsValue ... 172
Qualify ... 173
Rem ... 174
Rename ... 175
Search ... 176
Section ... 177
Select ... 178
Set ... 181
Sleep ... 181
SQL ... 181
SQLColumns ... 182

SQLTables 183
SQLTypes 184
Star 185
Store 186
Table/Tables 188
Tag 189
Trace 189
Unmap 190
Unqualify 191
Untag 191
2.6 Répertoire de travail 192
Répertoire de travail de Qlik Sense Desktop 192
Répertoire de travail de Qlik Sense 192
2 Utilisation des variables dans l'éditeur de chargement de données 193
2.7 Vue d'ensemble 193
2.8 Définition d'une variable 193
2.9 Suppression d'une variable 194
2.10 Chargement d'une valeur de variable comme valeur de champ 194
2.11 Calcul des variables 194
2.12 Variables système 195
Vue d'ensemble des variables système 195
CreateSearchIndexOnReload 198
HidePrefix 199
HideSuffix 199
Include 199
OpenUrlTimeout 201
StripComments 201
Verbatim 201
2.13 Variables de manipulation des valeurs 202
Vue d'ensemble des variables de manipulation des valeurs 202
NullDisplay 202
NullInterpret 203
NullValue 203
OtherSymbol 203
2.14 Variables d'interprétation des nombres 204
Formatage de la devise 204
Formatage des nombres 204
Formatage de l'heure 205
BrokenWeeks 206
DateFormat 207
DayNames 213
DecimalSep 218
FirstWeekDay 221
LongDayNames 225
LongMonthNames 228
MoneyDecimalSep 232
MoneyFormat 236

MoneyThousandSep 240
MonthNames 245
NumericalAbbreviation 250
ReferenceDay 251
ThousandSep 256
TimeFormat 262
TimestampFormat 262
2.15 Variables Direct Discovery 265
Variables système Direct Discovery 265
Variables de la fonction Bandes de requête de Teradata 267
Direct Discovery Variables de caractère 267
Variables d'interprétation des nombres Direct Discovery 268
2.16 Variables d'erreur 269
Vue d'ensemble des variables d'erreur 269
ErrorMode 270
ScriptError 270
ScriptErrorCount 271
ScriptErrorList 272
2 Expressions de script 273
3 Expressions de graphique 274
3.1 Définition de l'étendue d'une agrégation 274
3.2 Analyse d'ensembles 277
Expressions d'ensemble 277
Exemples 278
Ensembles naturels 278
Identificateurs d'ensemble 281
Opérateurs d'ensemble 282
Modificateurs d'ensemble 283
Expressions d'ensemble internes et externes 306
Didacticiel - Création d'une expression d'ensemble 308
Syntaxe des expressions d'ensemble 318
3.3 Syntaxe générale pour les expressions de graphique 319
3.4 Syntaxe générale pour les agrégations 319
4 Opérateurs 320
4.1 Opérateurs de bits 320
4.2 Opérateurs logiques 321
4.3 Opérateurs mathématiques 321
4.4 Opérateurs relationnels 322
4.5 Opérateurs de chaîne 324
& 324
like 324
5 Fonctions de script et de graphique 325
5.1 Connexions analytiques pour les extensions côté serveur (SSE) 325
5.2 Fonctions d'agrégation 325
Utilisation des fonctions d'agrégation dans un script de chargement de données 326
Utilisation des fonctions d'agrégation dans les expressions de graphique 326

Mode de calcul des agrégations 326
Agrégation de champs clés 326
Fonctions d'agrégation de base 327
Fonctions d'agrégation de décompte 351
Fonctions d'agrégation financières 369
Fonctions d'agrégation statistiques 391
Fonctions de test statistique 461
Fonctions d'agrégation de chaînes 529
Fonctions de dimension synthétique 542
Agrégations imbriquées 545
5.3 Aggr - fonction de graphique 545
Exemples : Expressions de graphique via Aggr 548
5.4 Fonctions de couleur 551
Fonctions de couleur prédéfinies 553
ARGB 554
RGB 554
HSL 556
5.5 Fonctions conditionnelles 557
Vue d'ensemble des fonctions conditionnelles 557
alt 558
class 559
coalesce 561
if 562
match 565
mixmatch 568
pick 571
wildmatch 572
5.6 Fonctions de décompte 575
Vue d'ensemble des fonctions de décompte 575
autonumber 576
autonumberhash128 578
autonumberhash256 580
IterNo 582
RecNo 583
RowNo 584
RowNo - fonction de graphique 586
5.7 Fonctions de date et heure 588
Vue d'ensemble des fonctions de date et heure 588
addmonths 597
addyears 607
age 615
converttolocaltime 617
day 619
dayend 626
daylightsaving 634
dayname 634
daynumberofquarter 636
daynumberofyear 643

daystart 649
firstworkdate 657
GMT 659
hour 663
inday 667
indaytotime 675
inlunarweek 686
inlunarweektodate 698
inmonth 709
inmonths 717
inmonthstodate 731
inmonthtodate 744
inquarter 755
inquartertodate 768
inweek 781
inweektodate 797
inyear 811
inyeartodate 824
lastworkdate 837
localtime 847
lunarweekend 848
lunarweekname 860
lunarweekstart 873
makedate 885
maketime 892
makeweekdate 899
minute 907
month 912
monthend 919
monthname 928
monthsend 936
monthsname 949
monthsstart 963
monthstart 976
networkdays 986
now 996
quarterend 1003
quartername 1017
quarterstart 1029
second 1041
setdateyear 1046
setdateyearmonth 1048
timezone 1050
today 1050
UTC 1056
week 1056
weekday 1074
weekend 1083

weekname 1094
weekstart 1109
weekyear 1121
year 1130
yearend 1137
yearname 1149
yearstart 1162
yeartodate 1174
5.8 Fonctions exponentielles et logarithmiques 1190
5.9 Fonctions de champ 1191
Fonctions de nombre 1191
Fonctions de champ et de sélection 1192
GetAlternativeCount - fonction de graphique 1192
GetCurrentSelections - fonction de graphique 1193
GetExcludedCount - fonction de graphique 1195
GetFieldSelections - fonction de graphique 1196
GetNotSelectedCount - fonction de graphique 1198
GetObjectDimension - fonction de graphique 1199
GetObjectField - fonction de graphique 1200
GetObjectMeasure - fonction de graphique 1200
GetPossibleCount - fonction de graphique 1201
GetSelectedCount - fonction de graphique 1203
5.10 Fonctions de fichier 1204
Vue d'ensemble des fonctions de fichier 1204
Attribute 1206
ConnectString 1213
FileBaseName 1214
FileDir 1214
FileExtension 1215
FileName 1215
FilePath 1215
FileSize 1216
FileTime 1217
GetFolderPath 1218
QvdCreateTime 1219
QvdFieldName 1220
QvdNoOfFields 1220
QvdNoOfRecords 1221
QvdTableName 1222
5.11 Fonctions financières 1223
Vue d'ensemble des fonctions financières 1224
BlackAndSchole 1224
FV 1225
nPer 1226
Pmt 1227
PV 1228
Rate 1229
5.12 Fonctions de formatage 1230

Vue d'ensemble des fonctions de formatage 1230
ApplyCodepage 1231
Date 1232
Dual 1234
Interval 1236
Money 1237
Num 1238
Time 1240
Timestamp 1242
5.13 Fonctions numériques générales 1243
Vue d'ensemble des fonctions numériques générales 1243
Fonctions de combinaison et de permutation 1244
Fonctions modulo 1244
Fonctions de parité 1244
Fonction d'arrondi 1245
BitCount 1245
Ceil 1245
Combin 1247
Div 1247
Even 1248
Fabs 1248
Fact 1249
Floor 1249
Fmod 1250
Frac 1251
Mod 1252
Odd 1253
Permut 1253
Round 1254
Sign 1256
5.14 Fonctions géospatiales 1256
Vue d'ensemble des fonctions géospatiales 1256
GeoAggrGeometry 1258
GeoBoundingBox 1259
GeoCountVertex 1260
GeoGetBoundingBox 1260
GeoGetPolygonCenter 1261
GeoInvProjectGeometry 1261
GeoMakePoint 1262
GeoProject 1263
GeoProjectGeometry 1263
GeoReduceGeometry 1264
5.15 Fonctions d'interprétation 1265
Vue d'ensemble des fonctions d'interprétation 1266
Date# 1267
Interval# 1268
Money# 1269
Num# 1270

Text 1271
Time# 1271
Timestamp# 1272
5.16 Fonctions d'inter-enregistrements 1273
Fonctions de ligne 1274
Fonctions de colonne 1275
Fonctions de champ 1275
Fonctions de tableau croisé dynamique 1276
Fonctions d'inter-enregistrements utilisées dans le script de chargement de données 1277
Above - fonction de graphique 1277
Below - fonction de graphique 1282
Bottom - fonction de graphique 1286
Column - fonction de graphique 1290
Dimensionality - fonction de graphique 1292
Exists 1294
FieldIndex 1297
FieldValue 1299
FieldValueCount 1300
LookUp 1302
NoOfRows - fonction de graphique 1304
Peek 1306
Previous 1312
Top - fonction de graphique 1313
SecondaryDimensionality - fonction de graphique 1317
After - fonction de graphique 1318
Before - fonction de graphique 1319
First - fonction de graphique 1320
Last - fonction de graphique 1321
ColumnNo - fonction de graphique 1322
NoOfColumns - fonction de graphique 1323
5.17 Fonctions logiques 1324
5.18 Fonctions de mappage 1325
Vue d'ensemble des fonctions de mappage 1325
ApplyMap 1325
MapSubstring 1327
5.19 Fonctions mathématiques 1329
5.20 Fonctions NULL 1330
Vue d'ensemble des fonctions NULL 1330
EmptyIsNull 1330
IsNull 1331
NULL 1332
5.21 Fonctions de plage 1333
Fonctions de plage de base 1333
Fonctions de plage de décompte 1334
Fonctions de plage statistiques 1335
Fonctions de plage financières 1335
RangeAvg 1336
RangeCorrel 1338

RangeCount ... 1341
RangeFractile ... 1343
RangeIRR ... 1345
RangeKurtosis ... 1346
RangeMax ... 1347
RangeMaxString ... 1349
RangeMin ... 1350
RangeMinString ... 1353
RangeMissingCount ... 1354
RangeMode ... 1356
RangeNPV ... 1358
RangeNullCount ... 1359
RangeNumericCount ... 1360
RangeOnly ... 1362
RangeSkew ... 1363
RangeStdev ... 1364
RangeSum ... 1366
RangeTextCount ... 1368
RangeXIRR ... 1369
RangeXNPV ... 1370
5.22 Fonctions de classement et de clustering ... 1371
Fonctions de classement dans les graphiques ... 1371
Fonctions de clustering dans les graphiques ... 1372
Rank - fonction de graphique ... 1373
HRank - fonction de graphique ... 1377
Optimisation à l'aide de K-moyennes : Exemple dans le monde réel ... 1379
KMeans2D - fonction de graphique ... 1389
KMeansND - fonction de graphique ... 1402
KMeansCentroid2D - fonction de graphique ... 1415
KMeansCentroidND - fonction de graphique ... 1417
5.23 Fonctions de distribution statistiques ... 1418
Vue d'ensemble des fonctions de distribution statistiques ... 1418
BetaDensity ... 1421
BetaDist ... 1421
BetaInv ... 1421
BinomDist ... 1422
BinomFrequency ... 1422
BinomInv ... 1423
ChiDensity ... 1423
ChiDist ... 1424
ChiInv ... 1424
FDensity ... 1425
FDist ... 1425
FInv ... 1426
GammaDensity ... 1427
GammaDist ... 1427
GammaInv ... 1428
NormDist ... 1428

NormInv 1429
PoissonDist 1430
PoissonFrequency 1430
PoissonInv 1430
TDensity 1431
TDist 1431
TInv 1432
5.24 Fonctions de chaîne 1433
Vue d'ensemble des fonctions de chaîne 1433
Capitalize 1437
Chr 1437
Evaluate 1438
FindOneOf 1438
Hash128 1439
Hash160 1440
Hash256 1441
Index 1442
IsJson 1443
JsonGet 1444
JsonSet 1445
KeepChar 1446
Left 1447
Len 1447
LevenshteinDist 1448
Lower 1449
LTrim 1450
Mid 1451
Ord 1452
PurgeChar 1452
Repeat 1453
Replace 1454
Right 1455
RTrim 1456
SubField 1456
SubStringCount 1459
TextBetween 1460
Trim 1461
Upper 1462
5.25 Fonctions système 1462
Vue d'ensemble des fonctions système 1463
EngineVersion 1465
IsPartialReload 1465
ProductVersion 1465
StateName - fonction de graphique 1465
5.26 Fonctions de table 1466
Vue d'ensemble des fonctions de table 1466
FieldName 1468
FieldNumber 1469

NoOfFields 1469
NoOfRows 1470
5.27 Fonctions trigonométriques et hyperboliques 1470
6 Restrictions d'accès au système de fichiers 1473
6.1 Aspects liés à la sécurité lors d'une connexion à des connexions de données ODBC et OLE DB basées sur des fichiers 1473
6.2 Limitations inhérentes au mode standard 1473
Variables système 1474
Instructions de script normales 1475
Instructions de contrôle de script 1477
Fonctions de fichier 1477
Fonctions système 1480
6.3 Désactivation du mode standard 1480
Qlik Sense 1480
Qlik Sense Desktop 1480
6 Scriptage graphique 1482
6.4 Instructions de contrôle 1482
Vue d'ensemble des instructions de contrôle d'un modificateur de graphique 1482
Call 1484
Do..loop 1485
End 1486
Exit 1486
Exit script 1486
For..next 1487
For each..next 1488
If..then..elseif..else..end if 1491
Next 1492
Sub..end sub 1492
Switch..case..default..end switch 1493
To 1494
6.5 Préfixes 1494
Vue d'ensemble des préfixes d'un modificateur de graphique 1494
Add 1495
Replace 1495
6.6 Instructions normales 1496
Vue d'ensemble des instructions normales d'un modificateur de graphique 1496
Load 1497
Let 1502
Set 1503
Put 1503
HCValue 1504
7 Fonctions et instructions QlikView non prises en charge dans Qlik Sense 1506
7.1 Instructions de script non prises en charge dans Qlik Sense 1506
7.2 Fonctions non prises en charge dans Qlik Sense 1506
7.3 Préfixes non pris en charge dans Qlik Sense 1506
8 Fonctions et instructions déconseillées dans Qlik Sense 1507
8.1 Instructions de script déconseillées dans Qlik Sense 1507

8.2 Paramètres d'instruction de script déconseillés dans Qlik Sense 1507
8.3 Fonctions déconseillées dans Qlik Sense 1508
Qualificateur ALL 1509

1 Présentation de Qlik Sense

Qlik Sense est une plate-forme d'analyse de données. Qlik Sense vous permet d'analyser des données et d'effectuer des découvertes de données. Vous pouvez partager des connaissances et analyser des données en groupes et au niveau de différentes organisations. Qlik Sense vous permet de poser des questions et d'y répondre afin de développer vos propres cheminements de pensée et perspectives. Qlik Sense vous permet, à vous et à vos collègues, de prendre des décisions en collaboration.

1.1 Opportunités offertes par Qlik Sense

La plupart des produits d'informatique décisionnelle (BI, Business Intelligence) vous aident à répondre à des questions prédéfinies à l'avance. Mais qu'en est-t-il des questions complémentaires ? Celles qui surviennent après la lecture de votre rapport ou la consultation de votre visualisation ? Grâce à l'expérience associative de Qlik Sense, vous pouvez répondre aux questions qui s'enchaînent, en suivant votre propre cheminement de pensée. Qlik Sense vous offre la possibilité d'explorer vos données librement, en quelques clics, d'apprendre progressivement et de déterminer les étapes à suivre en fonction des résultats antérieurs.

1.2 Fonctionnement de Qlik Sense

Qlik Sense génère à la volée des vues présentant les informations. Qlik Sense ne nécessite aucun rapport prédéfini ni statique. Vous ne dépendez pas d'autres utilisateurs ; il vous suffit de cliquer pour apprendre. Chacun de vos clics est immédiatement enregistré par Qlik Sense et toutes les vues et visualisations Qlik Sense sont mises à jour dans l'application, présentant un nouvel ensemble de données et de visualisations calculé, propre à vos sélections.

Modèle d'application

Le déploiement et la gestion d'applications d'entreprise démesurées n'est plus nécessaire : vous créez vos propres applications Qlik Sense que vous pouvez réutiliser, modifier et partager avec d'autres utilisateurs. Le modèle d'application vous permet de poser la question logique suivante et d'y répondre, sans devoir consulter un expert à chaque rapport ou visualisation que vous voulez créer.

Expérience associative

Qlik Sense gère automatiquement toutes les relations figurant dans les données et vous présente les informations sous forme de métaphore **green/white/gray**. Vos sélections sont mises en évidence en vert, les données associées en blanc et les données exclues (non associées) en gris. Ce retour d'information instantané vous permet de réfléchir à de nouvelles questions et de poursuivre l'exploration et la découverte.

Collaboration et mobilité

Qlik Sense vous permet de collaborer davantage avec vos collègues, à tout moment et en tout lieu. En effet, toutes les fonctionnalités de Qlik Sense, y compris la collaboration et l'expérience associative, sont disponibles sur les appareils mobiles. Grâce à Qlik Sense, vous pouvez poser des questions, y répondre et poser des questions de suivi à vos collègues, quel que soit l'endroit où vous vous trouvez.

1.3 Déploiement de Qlik Sense

Il existe deux versions de Qlik Sense à déployer, Qlik Sense Desktop et Qlik Sense Enterprise.

Qlik Sense Desktop

Il s'agit d'une version mono-utilisateur simple à installer, qui s'emploie généralement sur un ordinateur local.

Qlik Sense Enterprise

Cette version permet de déployer des sites Qlik Sense. Un site est une collection de machines serveur connectées à un référentiel logique commun ou nœud central.

1.4 Procédures d'administration et de gestion d'un site Qlik Sense

Avec la console Console de gestion Qlik, vous avez la possibilité de configurer, de gérer et de surveiller des sites Qlik Sense de manière simple et intuitive. Vous pouvez gérer des licences, des règles de sécurité et d'accès, configurer des nœuds et des connexions de sources de données, et synchroniser des contenus et des utilisateurs entre autres activités et ressources.

1.5 Extension de Qlik Sense et adaptation à vos propres besoins

Qlik Sense vous propose des API et des kits SDK flexibles, qui vous permettront de développer vos propres extensions ainsi que d'adapter et d'intégrer Qlik Sense à différentes fins, notamment aux fins suivantes :

Création d'extensions et d'applications composites

Cette section décrit des procédures de développement Web à l'aide de scripts JavaScript en vue de créer des extensions correspondant à des visualisations personnalisées dans les applications Qlik Sense. Vous pouvez aussi utiliser les API d'applications composites (mashups) pour élaborer des sites Web avec du contenu Qlik Sense.

Création de clients

Vous pouvez créer des clients dans .NET et incorporer des objets Qlik Sense dans vos propres applications. Vous avez par ailleurs la possibilité de développer des clients natifs dans n'importe quel langage de programmation compatible avec les communications WebSocket en utilisant le protocole client Qlik Sense.

Création d'outils serveur

Les API de services et d'annuaires d'utilisateurs vous permettent de créer votre propre outil d'administration et de gestion de sites Qlik Sense.

Connexion à d'autres sources de données

Créez des connecteurs Qlik Sense destinés à récupérer des données issues de sources personnalisées.

2 Vue d'ensemble de la syntaxe d'un script

2.1 Introduction à la syntaxe de script

Dans un script sont définis le nom de la source de données, les noms des tables et les noms des champs inclus dans la logique. Les champs figurant dans la définition des droits d'accès y sont également spécifiés. Un script se compose d'un certain nombre d'instructions qui sont exécutées de manière consécutive.

La syntaxe de ligne de commande et la syntaxe de script de Qlik Sense sont décrites dans une notation appelée Backus-Naur Formalism ou code BNF.

Les premières lignes de code sont déjà générées lorsqu'un nouveau fichier Qlik Sense est créé. Les valeurs par défaut de ces variables d'interprétation des nombres proviennent des paramètres régionaux du système d'exploitation.

Le script se compose d'un certain nombre d'instructions de script et de mots-clés qui sont exécutés de manière consécutive. Toutes les instructions de script doivent se terminer par un point-virgule, soit **;**.

Vous pouvez utiliser des expressions et des fonctions dans les instructions **LOAD** afin de transformer les données chargées.

Il est possible d'utiliser une instruction **LOAD** dans un fichier de table délimité par des virgules, des tabulations ou des points-virgules. Par défaut, une instruction **LOAD** charge tous les champs du fichier.

Les bases de données générales sont accessibles via les connecteurs de base de données ODBC ou OLE DB. Des instructions en code SQL standard y sont utilisées. La syntaxe SQL acceptée varie en fonction du pilote ODBC installé.

De plus, vous pouvez accéder à d'autres sources de données à l'aide de connecteurs personnalisés.

2.2 Présentation du code BNF (formalisme Backus-Naur)

La syntaxe de ligne de commande et la syntaxe de script de Qlik Sense sont décrites dans une notation appelée Backus-Naur Formalism, également connue sous l'abréviation BNF.

La table suivante présente une liste de symboles utilisés dans le code BNF, avec une description de leur mode d'interprétation :

Symboles

Symbole	Description
|	OR (OU) logique : les symboles qui se trouvent de l'un ou l'autre côté peuvent être utilisés.
( )	Parenthèses définissant la priorité : utilisées pour structurer la syntaxe BNF.
[ ]	Crochets : les éléments qu'ils contiennent sont facultatifs.

Symbole	Description
{ }	Accolades : les éléments qu'elles contiennent peuvent être répétés zéro ou plusieurs fois.
Symbole	Catégorie syntaxique non terminale, pouvant être divisée en d'autres symboles. Il peut s'agir, par exemple, de composés des éléments ci-dessus, d'autres symboles non terminaux, de chaînes textuelles, etc.
::=	Marque le début d'un bloc qui définit un symbole.
LOAD	Symbole terminal qui consiste en une chaîne textuelle. Doit être écrit tel quel dans le script.

Tous les symboles terminaux sont imprimés dans une police en caractères gras (**bold face**). Par exemple, le symbole (doit être interprété comme une parenthèse définissant la priorité tandis que **(** doit être interprété comme un caractère devant figurer dans le script.

Exemple :

La description de l'instruction alias est la suivante :

alias fieldname **as** aliasname { , fieldname **as** aliasname}

Elle doit être interprétée comme la chaîne textuelle "alias", suivie d'un nom de champ arbitraire, suivi de la chaîne textuelle "as", suivie d'un nom d'alias arbitraire. Il est possible d'indiquer autant de combinaisons supplémentaires que l'on veut de "fieldname as alias", séparées par des virgules.

Les instructions suivantes sont correctes :

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

Les instructions suivantes ne sont pas correctes :

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

2 Instructions de script et mots-clés

Les scripts Qlik Sense se composent d'un certain nombre d'instructions. Une instruction peut désigner soit une instruction de script normale, soit une instruction de contrôle de script. Certaines instructions peuvent être précédées de préfixes.

Les instructions normales servent généralement à manipuler des données d'une manière ou d'une autre. Ces instructions peuvent être écrites sur autant de lignes de script que nécessaire et doivent toujours se terminer par un point-virgule « ; ».

Les instructions de contrôle sont généralement utilisées pour contrôler le flux de l'exécution du script. Chaque clause d'une instruction de contrôle doit tenir sur une ligne de script et peut se terminer par un point-virgule ou une fin de ligne.

Il est possible d'appliquer des préfixes aux instructions normales pertinentes mais jamais aux instructions de contrôle. Les préfixes **when** et **unless** peuvent toutefois être utilisés comme suffixes pour quelques clauses d'instructions de contrôle bien précises.

Dans le sous-chapitre suivant, vous trouverez une liste alphabétique de toutes les instructions de script, de toutes les instructions de contrôle et de tous les préfixes.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

2.3 Instructions de contrôle de script

Les scripts Qlik Sense se composent d'un certain nombre d'instructions. Une instruction peut désigner soit une instruction de script normale, soit une instruction de contrôle de script.

Les instructions de contrôle sont généralement utilisées pour contrôler le flux de l'exécution du script. Chaque clause d'une instruction de contrôle doit tenir sur une ligne de script et peut se terminer par un point-virgule ou une fin de ligne.

Les préfixes ne s'appliquent jamais aux instructions de contrôle, à l'exception des préfixes **when** et **unless** qui sont compatibles avec certaines instructions.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules.

Vue d'ensemble des instructions de contrôle de script

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Call

L'instruction de contrôle **call** appelle une sous-routine qui doit être définie par une instruction **sub** précédente.

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

Do..loop

L'instruction de contrôle **do..loop** est une construction d'itération de script qui exécute une ou plusieurs instructions jusqu'à ce qu'une condition logique soit remplie.

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

Exit script

Cette instruction de contrôle arrête l'exécution du script. Elle peut être insérée n'importe où dans le script.

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

For each ..next

L'instruction de contrôle **for each..next** est une construction d'itération de script qui exécute une ou plusieurs instructions pour chaque valeur d'une liste de valeurs séparées par des virgules. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la liste.

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

For..next

L'instruction de contrôle **for..next** est une construction d'itération de script avec compteur. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la variable du compteur entre les limites inférieure et supérieure spécifiées.

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

If..then

L'instruction de contrôle **if..then** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon une ou plusieurs conditions logiques.

> *Comme l'instruction **if..then** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**if..then**, **elseif..then**, **else** et **end if**) ne peut s'étendre sur plusieurs lignes.*

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

```
end if
```

Sub

L'instruction de contrôle **sub..end sub** définit une sous-routine qui peut être appelée à partir d'une instruction **call**.

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

Switch

L'instruction de contrôle **switch** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon la valeur d'une expression.

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

Call

L'instruction de contrôle **call** appelle une sous-routine qui doit être définie par une instruction **sub** précédente.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
name	Nom de la sous-routine.
paramlist	Liste des paramètres à envoyer à la sous-routine, séparés par des virgules. Chaque élément de la liste peut correspondre à un nom de champ, une variable ou une expression arbitraire.

La sous-routine appelée par une instruction **call** doit être définie auparavant dans l'exécution du script par une instruction **sub**.

Les paramètres sont copiés dans la sous-routine et, si le paramètre de l'instruction **call** désigne une variable au lieu d'une expression, il est recopié et supprimé à la fermeture de la sous-routine.

Limitations :

- Comme l'instruction **call** est une instruction de contrôle et qu'elle se termine donc soit par un point-virgule, soit par un caractère de fin de ligne, elle ne doit pas s'étendre sur plusieurs lignes.
- Lorsque vous définissez une sous-routine avec `Sub..end sub` à l'intérieur d'une instruction de contrôle, par exemple `if..then`, vous pouvez uniquement appeler la sous-routine depuis la même instruction de contrôle.

Exemple :

Cet exemple répertorie tous les fichiers liés à Qlik dans un dossier et ses sous-dossiers, et enregistre les informations des fichiers dans une table. On suppose que vous avez créé une connexion de données appelée Apps avec le dossier.

La sous-routine DoDir est appelée avec la référence au dossier, 'lib://Apps', en tant que paramètre. La sous-routine contient un appel récursif `Call DoDir (Dir)` qui permet à la fonction de rechercher des fichiers dans les sous-dossiers de façon récursive.

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

Do..loop

L'instruction de contrôle **do..loop** est une construction d'itération de script qui exécute une ou plusieurs instructions jusqu'à ce qu'une condition logique soit remplie.

Syntaxe :

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

> *Comme l'instruction **do..loop** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**do**,**exit do** et **loop**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.
while / until	Les clauses conditionnelles **while** ou **until** ne doivent figurer qu'une fois dans une instruction **do..loop**, soit après **do**, soit après **loop**. Chaque condition n'est interprétée que la première fois, mais elle est évaluée à chaque fois que le script la rencontre dans la boucle.
exit do	Si une clause **exit do** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **loop** indiquant la fin de la boucle. Il est possible de rendre une clause **exit do** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

Exemple :

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

End

Le mot-clé de script **End** permet de fermer les clauses **If**, **Sub** et **Switch**.

Exit

Le mot-clé de script **Exit** fait partie de l'instruction **Exit Script**, mais il peut aussi être employé pour quitter les clauses **Do**, **For** ou **Sub**.

Exit script

Cette instruction de contrôle arrête l'exécution du script. Elle peut être insérée n'importe où dans le script.

Syntaxe :

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

Comme l'instruction **exit script** est une instruction de contrôle et qu'elle se termine donc soit par un point-virgule, soit par un caractère de fin de ligne, elle ne doit pas s'étendre sur plusieurs lignes.

Arguments :

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat True ou False.
when / unless	Il est possible de rendre une instruction **exit script** conditionnelle par l'utilisation facultative d'une clause **when** ou **unless**.

Exemples :

```
//Exit script
Exit Script;

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

For..next

L'instruction de contrôle **for..next** est une construction d'itération de script avec compteur. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la variable du compteur entre les limites inférieure et supérieure spécifiées.

Syntaxe :

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

Les expressions *expr1*, *expr2* et *expr3* ne sont évaluées que la première fois que le script entre dans la boucle. Il est possible de modifier la valeur de la variable counter à l'aide d'instructions placées à l'intérieur de la boucle, mais ce n'est pas une bonne méthode de programmation.

Si une clause **exit for** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **next** indiquant la fin de la boucle. Il est possible de rendre une clause **exit for** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

*Comme l'instruction **for..next** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**for..to..step**,**exit for** et **next**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
counter	Nom de variable. Si l'argument *counter* est spécifié après **next**, il doit s'agir du même nom de variable que celui qui se trouve après le **for** correspondant.
expr1	Expression qui détermine la première valeur de la variable *counter* pour laquelle la boucle doit être exécutée.
expr2	Expression qui détermine la dernière valeur de la variable *counter* pour laquelle la boucle doit être exécutée.
expr3	Expression qui détermine la valeur de l'incrément de la variable *counter* lors de chaque exécution de la boucle.
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Example 1: Chargement d'une séquence de fichiers

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

      LOAD * from file$(a).csv;

next
```

Example 2: Chargement d'un nombre aléatoire de fichiers

Dans cet exemple, supposons les fichiers de données *x1.csv, x3.csv, x5.csv, x7.csv* et *x9.csv*. Le chargement est arrêté en un point aléatoire à l'aide de la condition `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

L'instruction de contrôle **for each..next** est une construction d'itération de script qui exécute une ou plusieurs instructions pour chaque valeur d'une liste de valeurs séparées par des virgules. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la liste.

Syntaxe :

Une syntaxe spéciale permet de générer des listes comprenant les noms des fichiers et des répertoires contenus dans le répertoire actif.

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

Arguments :

Arguments

Argument	Description
var	Nom de variable de script qui prendra une nouvelle valeur à partir de la liste lors de chaque exécution de la boucle. Si l'argument **var** est spécifié après **next**, il doit s'agir du même nom de variable que celui qui se trouve après le **for each** correspondant.

Il est possible de modifier la valeur de la variable **var** à l'aide d'instructions placées à l'intérieur de la boucle, mais ce n'est pas une bonne méthode de programmation.

Si une clause **exit for** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **next** indiquant la fin de la boucle. Il est possible de rendre une clause **exit for** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

*Comme l'instruction **for each..next** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**for each**,**exit for** et **next**) ne peut s'étendre sur plusieurs lignes.*

Syntaxe :

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

Arguments

Argument	Description
constant	Tout nombre ou toute chaîne. Veuillez noter qu'une chaîne écrite directement dans le script doit être placée entre guillemets simples. Une chaîne non mise entre guillemets simples est interprétée comme une variable ; la valeur de la variable lui est ensuite appliquée. Il est inutile de placer les nombres entre guillemets simples.
expression	Expression arbitraire.
mask	Masque de nom de fichier ou de dossier pouvant inclure n'importe quel caractère de nom de fichier valide, ainsi que les caractères génériques standard, ***** et **?**. Vous pouvez utiliser des chemins d'accès absolus ou des chemins d'accès lib://.
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.
filelist mask	Cette syntaxe génère une liste de tous les fichiers, séparés par des virgules, qui se trouvent dans le répertoire actif et qui correspondent au masque de nom de fichier. *Cet argument prend uniquement en charge les connexions aux bibliothèques en mode standard.*

Argument	Description
dirlist mask	Cette syntaxe génère une liste de tous les dossiers, séparés par des virgules, qui se trouvent dans le dossier actif et qui correspondent au masque de nom de dossier. *Cet argument prend uniquement en charge les connexions aux bibliothèques en mode standard.*
fieldvaluelist mask	Cette syntaxe itère au sein des valeurs d'un champ déjà chargé dans Qlik Sense.

*Le Qlik Connecteurs de fournisseurs de stockage Web et les autres connexions DataFiles ne prennent pas en charge les masques de filtre utilisant les caractères génériques (***** et **?**).*

Example 1: Chargement d'une liste de fichiers

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

Example 2: Création d'une liste de fichiers sur un disque

Dans cet exemple, la liste de tous les fichiers relatifs à Qlik Sense sont chargés dans un dossier.

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

Example 3: Itération au sein des valeurs d'un champ

Cet exemple itère au sein de la liste des valeurs chargées de champ FIELD et génère un nouveau champ, NEWFIELD. Pour chaque valeur de FIELD, deux enregistrements NEWFIELD sont créés.

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

La table résultante a l'aspect suivant :

Example table

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

NEWFIELD
three-1
three-2

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

L'instruction de contrôle **if..then** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon une ou plusieurs conditions logiques.

Les instructions de contrôle sont généralement utilisées pour contrôler le flux d'exécution du script. Dans une expression de graphique, utilisez plutôt la fonction conditionnelle **if**.

Syntaxe :

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

Comme l'instruction **if..then** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**if..then**, **elseif..then**, **else** et **end if**) ne peut s'étendre sur plusieurs lignes.

Arguments :

Arguments

Argument	Description
condition	Expression logique qui peut être évaluée comme True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Example 1:

```
if a=1 then

      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;

end if
```

Example 2:

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

Example 3:

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

Next

Le mot-clé de script **Next** permet de fermer les boucles **For**.

Sub..end sub

L'instruction de contrôle **sub..end sub** définit une sous-routine qui peut être appelée à partir d'une instruction **call**.

Syntaxe :

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

Les arguments sont copiés dans la sous-routine et, si les paramètres réels correspondants de l'instruction **call** constituent un nom de variable, ils sont recopiés et supprimés à la fermeture de la sous-routine.

Si une sous-routine comporte plus de paramètres formels que ceux réellement transmis par une instruction **call**, les paramètres supplémentaires sont initialisés sur la valeur NULL et peuvent être utilisés comme variables locales dans la sous-routine.

Arguments :

Arguments

Argument	Description
name	Nom de la sous-routine.
paramlist	Liste de noms de variables séparés par des virgules et définissant les paramètres formels de la sous-routine. Ceux-ci peuvent être utilisés comme n'importe quelle variable au sein de la sous-routine.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Limitations :

- Comme l'instruction **sub** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses deux clauses possibles (**sub** et **end sub**) ne peut s'étendre sur plusieurs lignes.
- Lorsque vous définissez une sous-routine avec `Sub..end sub` à l'intérieur d'une instruction de contrôle, par exemple `if..then`, vous pouvez uniquement appeler la sous-routine depuis la même instruction de contrôle.

Example 1:

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

Example 2: - transfert de paramètres

```
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)
```

Le résultat de l'exemple ci-dessus est que, localement, au sein de la sous-routine, A sera initialisé sur 1, B sur 4 et C sur NULL.

Au moment de quitter la sous-routine, la variable globale A obtiendra la valeur 2 (recopiée à partir de la sous-routine). Le deuxième paramètre réel (X+1)*2 ne sera pas recopié, car il ne s'agit pas d'une variable. Enfin, la variable globale C ne sera pas affectée par l'appel de sous-routine.

Switch..case..default..end switch

L'instruction de contrôle **switch** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon la valeur d'une expression.

Syntaxe :

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

*Comme l'instruction **switch** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**switch**, **case**, **default** et **end switch**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
expression	Expression arbitraire.
valuelist	Liste de valeurs séparées par des virgules à laquelle la valeur de l'expression sera comparée. L'exécution du script se poursuit avec les instructions du premier groupe rencontré qui comporte dans l'argument valuelist une valeur égale à la valeur de l'expression. Chaque valeur de l'argument valuelist peut désigner une expression arbitraire. Si aucune valeur correspondante n'est trouvée dans une clause **case**, les instructions figurant dans la clause **default** (si celle-ci est spécifiée) sont exécutées.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Exemple :

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

To

Le mot-clé de script **To** s'utilise dans plusieurs instructions de script.

2.4 Préfixes de script

Il est possible d'appliquer des préfixes aux instructions normales pertinentes mais jamais aux instructions de contrôle. Les préfixes **when** et **unless** peuvent toutefois être utilisés comme suffixes pour quelques clauses d'instructions de contrôle bien précises.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

Vue d'ensemble des préfixes de script

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Add

Il est possible d'ajouter le préfixe **Add** à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier qu'il faut ajouter des enregistrements à une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Add** peut également être utilisé dans une instruction **Map**.

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

Buffer

Il est possible de créer et de gérer automatiquement des fichiers QVD à l'aide du préfixe **buffer**. Ce préfixe peut être utilisé dans la plupart des instructions **LOAD** et **SELECT** du script. Il indique que des fichiers QVD sont utilisés pour mettre en cache/mémoire tampon le résultat de l'instruction.

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

Concatenate

Si deux tables qui doivent être concaténées comportent des ensembles de champs différents, il est tout de même possible de forcer la concaténation des deux tables à l'aide du préfixe **Concatenate**.

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

Crosstable

Le préfixe de chargement **crosstable** est utilisé pour transposer des données structurées de « tableau croisé » ou de « tableau croisé dynamique ». Les données structurées de cette manière sont fréquentes lorsque vous travaillez avec des sources de feuilles de calcul. Le résultat et l'objectif du préfixe de chargement **crosstable** sont de transposer ces structures dans un équivalent de tableau avec des colonnes standard, car cette structure est généralement mieux adaptée à l'analyse dans Qlik Sense.

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

First

Le préfixe **First** associé à une instruction **LOAD** ou **SELECT (SQL)** sert à charger un nombre d'enregistrements maximal défini à partir d'une table de source de données.

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

Generic

Le préfixe de chargement **Generic** permet la conversion des données modélisées entité-attribut-valeur (EAV) en une structure de table relationnelle normalisée standard. La modélisation EAV est également appelée « modélisation de données générique » ou « schéma ouvert ».

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

Hierarchy

Le préfixe **hierarchy** permet de transformer une table de hiérarchies parent-enfant en table utile dans un modèle de données Qlik Sense. Vous pouvez l'insérer devant une instruction **LOAD** ou **SELECT**. Il utilise le résultat de l'instruction de chargement comme entrée pour une transformation de table.

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

HierarchBelongsTo

Ce préfixe permet de transformer une table de hiérarchies parent-enfant en table utile dans un modèle de données Qlik Sense. Vous pouvez l'insérer devant une instruction **LOAD** ou **SELECT**. Il utilise le résultat de l'instruction de chargement comme entrée pour une transformation de table.

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

Inner

Les préfixes **join** et **keep** peuvent être précédés du préfixe **inner**.

Utilisé avant **join**, il spécifie l'utilisation d'une jointure interne. De ce fait, la table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans les deux tables. Utilisé avant **keep**, il indique que les deux tables de données brutes doivent être réduites à leur intersection commune avant d'être stockées dans Qlik Sense.

.

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

IntervalMatch

Le préfixe **IntervalMatch** permet de créer une table faisant correspondre des valeurs numériques discrètes à un ou plusieurs intervalles numériques et, de manière facultative, faisant correspondre les valeurs d'une ou de plusieurs clés supplémentaires.

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

Join

Le préfixe **join** permet de joindre la table chargée à une table nommée existante ou à la dernière table de données créée.

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

Keep

Le préfixe **keep** est semblable au préfixe **join**. Tout comme le préfixe **join**, il compare la table chargée à une table nommée existante ou à la dernière table de données créée. Cependant, au lieu de joindre la table chargée à une table existante, il a pour effet de réduire une ou les deux tables avant qu'elles ne soient stockées dans Qlik Sense, en fonction de l'intersection des données des tables. La comparaison

effectuée équivaut à une jonction naturelle entre tous les champs communs, c.-à-d. de la même manière que dans une jonction correspondante. Cependant, les deux tables ne sont pas jointes et sont conservées dans Qlik Sense comme deux tables nommées distinctes.

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

Left

Les préfixes **Join** et **Keep** peuvent être précédés du préfixe **left**.

Utilisé avant **join**, il spécifie l'utilisation d'une jointure gauche. La table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans la première table. Utilisé avant **keep**, il indique que la deuxième table de données brutes doit être réduite à son intersection commune avec la première table avant d'être stockée dans Qlik Sense.

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

Mapping

Le préfixe **mapping** permet de créer une table de mappage pouvant servir, par exemple, à remplacer des valeurs de champ et des noms de champ lors de l'exécution du script.

```
Mappage  ( loadstatement | selectstatement )
```

Merge

Le préfixe **Merge** peut être ajouté à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier que la table chargée doit être fusionnée dans une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel.

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

NoConcatenate

Le préfixe **NoConcatenate** oblige deux tables chargées aux ensembles de champs identiques à être traitées comme deux tables internes distinctes. Sinon, elles seraient automatiquement concaténées.

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

Outer

Le préfixe explicite **Join** peut être précédé du préfixe **Outer** pour spécifier une jointure externe. Dans une jointure externe, toutes les combinaisons entre les deux tables sont générées. De ce fait, la table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans une ou les deux tables. Le mot-clé **Outer** est facultatif et correspond au type de jointure par défaut utilisé lorsqu'un préfixe de jointure n'est pas spécifié.

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

Partial reload

Un chargement complet commence toujours par supprimer toutes les tables du modèle de données existant, puis exécute le script de chargement.

Un *Chargement partiel (page 101)* ne le fait pas. Au lieu de cela, il conserve toutes les tables du modèle de données, puis exécute uniquement les instructions**Load** et **Select** précédées d'un préfixe **Add**, **Merge** ou **Replace**. Les autres tables de données ne sont pas affectées par la commande. L'argument **only** indique que l'instruction doit être exécutée uniquement lors des chargements partiels et qu'elle doit être ignorée lors des chargements complets. Le tableau suivant synthétise l'exécution des instructions pour les chargements partiels et complets.

Replace

Le préfixe **Replace** peut être ajouté à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier que la table chargée doit remplacer une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Replace** peut également être utilisé dans une instruction **Map**.

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

Right

Les préfixes **Join** et **Keep** peuvent être précédés du préfixe **right**.

Utilisé avant **join**, il spécifie l'utilisation d'une jointure droite. La table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans la deuxième table. Utilisé avant **keep**, il indique que la première table de données brutes doit être réduite à son intersection commune avec la deuxième table avant d'être stockée dans Qlik Sense.

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

Sample

Le préfixe **sample** associé à une instruction **LOAD** ou **SELECT** permet de charger un échantillon aléatoire d'enregistrements à partir de la source de données.

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

Semantic

Il est possible de charger des tables contenant des relations entre des enregistrements à l'aide d'un préfixe **semantic**. Il peut s'agir, par exemple, d'auto-références au sein d'une table, où un enregistrement pointe vers un autre, tel qu'un parent, auquel il appartient ou qui est son prédécesseur.

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

Unless

Utilisé comme préfixe ou comme suffixe, **unless** permet de créer une clause conditionnelle qui détermine si une instruction ou une clause exit doit être évaluée ou pas. Il peut être considéré comme une alternative plus compacte à l'instruction complète **if..end if**.

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

When

Utilisé comme préfixe ou comme suffixe, **when** permet de créer une clause conditionnelle qui détermine si une instruction ou une clause exit doit être exécutée ou pas. Il peut être considéré comme une alternative plus compacte à l'instruction complète **if..end if**.

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

Add

Il est possible d'ajouter le préfixe **Add** à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier qu'il faut ajouter des enregistrements à une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Add** peut également être utilisé dans une instruction **Map**.

Pour que le chargement partiel fonctionne correctement, vous devez ouvrir l'application avec des données avant le déclenchement du chargement partiel.

Effectuez un chargement partiel via le bouton **Charger**. Vous pouvez également utiliser Qlik Engine JSON API.

Syntaxe :

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

```
Add [only] mapstatement
```

Lors d'un chargement normal (non partiel), la construction **Add LOAD** fonctionne comme une instruction **LOAD** normale. Les enregistrements seront générés et stockés dans une table.

Si le préfixe **Concatenate** est utilisé ou s'il existe une table avec le même ensemble de champs, les enregistrements seront ajoutés à la table existante correspondante. Sinon, la construction **Add LOAD** créera une table.

Un chargement partiel fera la même chose. La seule différence réside dans le fait que la construction **Add LOAD** ne créera jamais de table. Il existe toujours une table correspondante provenant de la précédente exécution de script à laquelle les enregistrements doivent être ajoutés.

La présence de doublons n'est pas vérifiée. Par conséquent, une instruction utilisant le préfixe **Add** inclut généralement soit un qualificateur distinct, soit une clause where conservant les doublons.

L'instruction **Add Map...Using** permet également d'effectuer le mappage pendant une exécution de script partielle.

Arguments :

Arguments

Argument	Description
only	Qualificateur facultatif indiquant que l'instruction doit être exécutée uniquement lors des chargements partiels. Lors des chargements normaux (non partiels), elle doit être ignorée.

Exemples et résultats :

Exemple	Résultat

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	Pendant un rechargement normal, les données sont chargées à partir du fichier *Persons.csv* et stockées dans la table Qlik Sense Tab1. Les données du fichier *NewPersons.csv* sont ensuite concaténées dans la même table Qlik Sense.
	
	Pendant un rechargement partiel, les données sont chargées à partir du fichier *NewPersons.csv* et ajoutées à la table Qlik Sense Tab1. La présence de doublons n'est pas vérifiée.

`Tab1:`
 	
`SQL SELECT Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM NewPersons.csv where not exists (Name);`
	Le programme vérifie l'absence de doublons en recherchant des occurrences de Name dans les données de table déjà chargées.
	
	Pendant un rechargement normal, les données sont chargées à partir du fichier *Persons.csv* et stockées dans la table Qlik Sense Tab1. Les données du fichier *NewPersons.csv* sont ensuite concaténées dans la même table Qlik Sense.
	
	Pendant un rechargement partiel, les données sont chargées à partir du fichier *NewPersons.csv*, lequel est ajouté à la table Qlik Sense Tab1. Le programme vérifie l'absence de doublons en recherchant des occurrences de Name dans les données de table déjà chargées.

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add Only LOAD Name, Number FROM NewPersons.csv where not exists (Name);`
	Pendant un rechargement normal, les données sont chargées à partir du fichier *Persons.csv* et stockées dans la table Qlik Sense Tab1. L'instruction de chargement du fichier *NewPersons.csv* est ignorée.
	
	Pendant un rechargement partiel, les données sont chargées à partir du fichier *NewPersons.csv*, lequel est ajouté à la table Qlik Sense Tab1. Le programme vérifie l'absence de doublons en recherchant des occurrences de Name dans les données de table déjà chargées.

Buffer

Il est possible de créer et de gérer automatiquement des fichiers QVD à l'aide du préfixe **buffer**. Ce préfixe peut être utilisé dans la plupart des instructions **LOAD** et **SELECT** du script. Il indique que des fichiers QVD sont utilisés pour mettre en cache/mémoire tampon le résultat de l'instruction.

Syntaxe :

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

Si aucune option n'est utilisée, le tampon (buffer) QVD créé par la première exécution du script est utilisé indéfiniment.

Le fichier de tampon est stocké dans le sous-dossier *Buffers*, généralement installé sous *C:\ProgramData\Qlik\Sense\Engine\Buffers* (installation serveur) ou *C:\Utilisateurs\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

Le nom du fichier QVD est un nom calculé, hachage hexadécimal de 160 bits de toute l'instruction **LOAD** ou **SELECT** qui suit et d'autres informations discriminantes. Autrement dit, cela signifie que le tampon QVD ne sera plus valide si l'instruction **LOAD** ou **SELECT** qui suit est modifiée.

Les tampons QVD sont normalement supprimés lorsqu'ils ne sont plus référencés nulle part lors d'une exécution de script complète dans l'application qui les a créés ou lorsque l'application qui les a créés n'existe plus.

Arguments :

Arguments

Argument	Description
incremental	L'option incremental permet de ne lire qu'une partie d'un fichier sous-jacent. La taille précédente du fichier est stockée dans l'en-tête XML du fichier QVD. Cette méthode s'avère particulièrement utile dans le cas des fichiers journaux. Tous les enregistrements chargés précédemment sont lus à partir du fichier QVD tandis que les nouveaux enregistrements ultérieurs sont lus à partir de la source d'origine, avant qu'un fichier QVD à jour ne soit créé. L'option incremental peut uniquement être utilisée avec des instructions **LOAD** et des fichiers texte. Il n'est pas possible d'utiliser un chargement incrémentiel lorsque les anciennes données sont modifiées ou supprimées.
stale [after] amount [(days | hours)]	amount est un nombre spécifiant la période. Cet argument admet l'utilisation de décimales. Si l'unité n'est pas précisée, les jours sont utilisés par défaut. L'option stale after est généralement utilisée avec des sources de base de données qui ne comportent pas d'horodatage simple pour les données d'origine. À la place, vous spécifiez de quand peut dater l'instantané QVD à utiliser. Une clause stale after spécifie simplement une période commençant à la date de création du tampon QVD et après laquelle celui-ci ne sera plus considéré comme valide. Avant l'expiration de ce délai, le tampon QVD est utilisé comme source de données et après ce moment, c'est la source de données d'origine. Le fichier de tampon QVD est alors mis à jour automatiquement et une nouvelle période débute.

Limitations :

Cette fonction présente de nombreuses limites, la plus notable étant qu'une instruction **LOAD** de fichier ou une instruction **SELECT** soit au cœur de toute instruction complexe.

Example 1:

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

Example 2:

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

Example 3:

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

Concatenate

`Concatenate` est un préfixe de chargement de script qui permet d'ajouter un ensemble de données à une table en mémoire qui existe déjà. Il est souvent utilisé pour ajouter différents ensembles de données transactionnelles à une seule table de faits centrale ou pour créer des ensembles de données de référence communs d'un type spécifique provenant de différentes sources. Sa fonctionnalité est similaire à celle d'un opérateur SQL UNION.

La table obtenue d'une opération `concatenate` contiendra l'ensemble de données d'origine avec les nouvelles lignes de données ajoutées au bas de cette table. Les tables source et cible peuvent contenir des champs différents. Lorsque ces champs sont différents, la table obtenue est élargie pour représenter le résultat combiné de l'ensemble des champs présents dans les deux tables source et cible.

Syntaxe :

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

Arguments

Argument	Description
`tablename`	Nom d'une table existante. La table nommée sera la cible de l'opération `Concatenate` et l'ensemble des enregistrements de données chargés seront ajoutés à cette table. Si le paramètre `tablename` n'est pas utilisé, la table cible sera la dernière table chargée avant cette instruction.
`loadstatement/selectstatement`	L'argument `loadstatement/selectstatement` qui suit l'argument `tablename` sera concaténé à la table spécifiée.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple de fonction

Exemple	Résultat
`Concatenate (Transactions) Load …. ;`	Les données chargées dans l'instruction LOAD sous le préfixe `Concatenate` seront ajoutées à la table en mémoire existante appelée `Transactions` (en supposant qu'une table appelée `Transactions` a été chargée avant ce point dans le script de chargement).

Exemple 1 – ajout de plusieurs ensembles de données à une table cible avec le préfixe de chargement Concatenate

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, vous allez charger deux scripts dans l'ordre séquentiel.

- Le premier script de chargement contient un ensemble de données initial avec des dates et des montants, envoyé à une table appelée `Transactions`.
- Le deuxième script contient les éléments suivants :
  - Un deuxième ensemble de données, ajouté à l'ensemble de données initial via le préfixe `Concatenate`. Cet ensemble de données comporte un champ supplémentaire, `type`, qui ne figure pas dans l'ensemble de données initial.
  - Le préfixe `Concatenate`.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Premier script de chargement

```
Transactions:
Load * Inline [

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- amount

Tableau de résultats du premier script de chargement

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

Le tableau montre l'ensemble de données initial.

Deuxième script de chargement

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous.

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

Résultats

Chargez les données et accédez à la feuille. Créez ce champ comme dimension :

- type

Tableau de résultats du deuxième script de chargement

id	date	amount	type
3750	08/30/2018	23.56	-

id	date	amount	type
3751	09/07/2018	556.31	-
3752	09/16/2018	5.75	-
3753	09/22/2018	125.00	-
3754	09/22/2018	484.21	-
3756	09/22/2018	59.18	-
3757	09/23/2018	177.42	-
3758	10/01/2018	164.27	Internal
3759	10/03/2018	384.00	External
3760	10/06/2018	25.82	Internal
3761	10/09/2018	312.00	Internal
3762	10/15/2018	4.56	Internal
3763	10/16/2018	90.24	Internal
3764	10/18/2018	19.32	External

Notez les valeurs nulles du champ `type` des sept premiers enregistrements chargés pour lesquels la valeur `type` n'a pas été définie.

Exemple 2 – ajout de plusieurs ensembles de données à une table cible via une concaténation implicite

Script de chargement et résultats

Vue d'ensemble

Un cas d'usage type de l'ajout implicite de données se produit lorsque vous chargez plusieurs fichiers de données structurées de la même manière et que vous souhaitez tous les ajouter à une table cible.

Par exemple, en utilisant `wildcards` dans des noms de fichier avec une syntaxe similaire à l'exemple suivant :

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

ou dans des boucles utilisant des constructions similaires à l'exemple suivant :

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

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

next file
```

La concaténation implicite se produira entre deux tables, quelles qu'elles soient, chargées comportant des champs portant le même nom, même si elles ne sont pas définies l'une par rapport à l'autre dans le script. Cela peut entraîner l'ajout involontaire de données à des tables. Si vous ne souhaitez pas qu'une table secondaire contenant des champs identiques soit ajoutée de cette manière, utilisez le préfixe de chargement `NoConcatenate`. Le renommage de la table à l'aide d'une autre balise de nom de table n'est pas suffisant pour empêcher la concaténation implicite. Pour plus d'informations, voir NoConcatenate (page 91).

Dans cet exemple, vous allez charger deux scripts dans l'ordre séquentiel.

- Le premier script de chargement contient un ensemble de données initial avec quatre champs, envoyé à une table appelée `Transactions`.
- Le deuxième script de chargement contient un ensemble de données comportant les mêmes champs que le premier ensemble de données.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Premier script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau de résultats du premier script de chargement

id	date	type	amount
3758	10/01/2018	Internal	164.27
3759	10/03/2018	External	384.00

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

Le tableau montre l'ensemble de données initial.

Deuxième script de chargement

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous.

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

Résultats

Chargez les données et accédez à la feuille.

Tableau de résultats du deuxième script de chargement

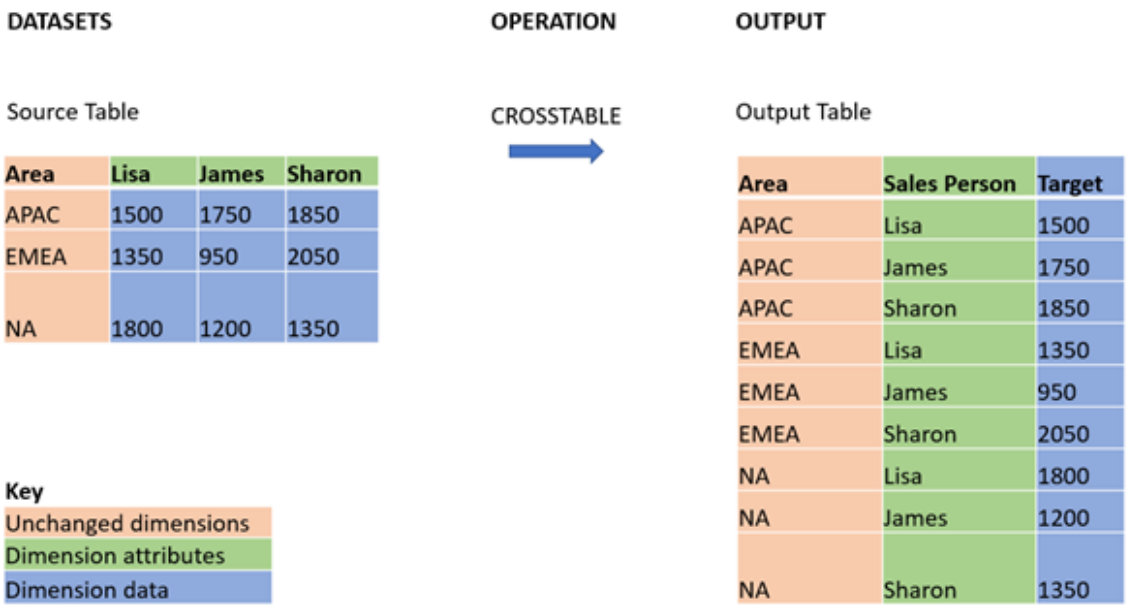
id	date	type	amount
3758	10/01/2018	Internal	164.27
3759	10/03/2018	External	384.00
3760	10/06/2018	Internal	25.82
3761	10/09/2018	Internal	312.00
3762	10/15/2018	Internal	4.56
3763	10/16/2018	Internal	90.24
3764	10/18/2018	External	19.32
3765	11/03/2018	Internal	129.40
3766	11/05/2018	External	638.50

Le deuxième ensemble de données a été implicitement concaténé à l'ensemble de données initial, car ils avaient des champs identiques.

Crosstable

Le préfixe de chargement **crosstable** est utilisé pour transposer des données structurées de « tableau croisé » ou de « tableau croisé dynamique ». Les données structurées de cette manière sont fréquentes lorsque vous travaillez avec des sources de feuilles de calcul. Le résultat et l'objectif du préfixe de chargement **crosstable** sont de transposer ces structures dans un équivalent de tableau avec des colonnes standard, car cette structure est généralement mieux adaptée à l'analyse dans Qlik Sense.

Exemple de données structurées sous forme de tableau croisé (crosstable) et sa structure équivalente après une transformation crosstable



Syntaxe :

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

Arguments

Argument	Description
attribute field name	Nom de champ de sortie souhaité décrivant la dimension orientée à l'horizontale à transposer (la ligne d'en-tête).
data field name	Nom de champ de sortie souhaité décrivant les données orientées à l'horizontale de la dimension à transposer (la matrice des valeurs de données sous la ligne d'en-tête).
n	Nombre de champs du qualificateur, ou dimensions inchangées, précédant le tableau à transformer au format générique. La valeur par défaut est 1.

Cette fonction de script est liée aux fonctions suivantes :

Fonctions associées

Fonction	Interaction
Generic (page 59)	Préfixe de chargement transformation qui prend un ensemble de données structuré au format entité-attribut-valeur et le transforme en une structure de table relationnelle standard, séparant chaque attribut rencontré dans un nouveau champ ou dans une nouvelle colonne de données.

Exemple 1 – Transformation de données sales pivotées (simple)

Scripts de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le premier script de chargement ci-dessous à un nouvel onglet.

Le premier script de chargement contient un ensemble de données auquel le préfixe de script `crosstable` sera appliqué ultérieurement, la section appliquant `crosstable` commenté. Cela signifie que la syntaxe de commentaire a été utilisée pour désactiver cette section dans le script de chargement.

Le deuxième script de chargement est le même que le premier, mais avec l'application de `crosstable` non commenté (activé via la suppression de la syntaxe de commentaire). Les scripts sont présentés de cette manière pour mettre en évidence la valeur de cette fonction de script dans la transformation des données.

Premier script de chargement (fonction non appliquée)

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

//Final:
//Load Product,
//Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
//Sales

//Resident tmpData;

//Drop Table tmpData;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau de résultats

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

Ce script permet la création d'un tableau croisé (crosstable) avec une colonne pour chaque mois et une ligne par produit. Dans ce format, ces données ne sont pas faciles à analyser. Il serait préférable d'avoir tous les nombres dans un champ et tous les mois dans un autre, à savoir, dans un tableau à trois colonnes. La section suivante explique comment effectuer cette transformation en tableau croisé (crosstable).

Deuxième script de chargement (fonction appliquée)

Décommentez le script en supprimant `//`. Le script de chargement devrait ressembler au suivant :

```
tmpData:
Crosstable (MonthText, Sales)
Load * inline [
Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021
A, 100, 98, 103, 63, 108, 82
B, 284, 279, 297, 305, 294, 292
C, 50, 53, 50, 54, 49, 51];

Final:
Load Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales

Resident tmpData;

Drop Table tmpData;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `Product`
- `Month`
- `Sales`

Tableau de résultats

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

Une fois le préfixe de script appliqué, le tableau croisé (crosstable) est transformé en tableau simple avec une colonne pour `Month` et une autre pour `Sales`. Cela améliore la lisibilité des données.

Exemple 2 - Transformation de données sales target pivotées en une structure de tableau verticale (intermédiaire)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée Targets.
- Préfixe de chargement `crosstable`, qui transpose les noms sales person pivotés dans un champ qui lui est propre, intitulé `Sales Person`.
- Données sales target associées, structurées dans un champ nommé `Target`.

Script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `Area`
- `Sales Person`

Ajoutez cette mesure :

```
=Sum(Target)
```

Tableau de résultats

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

Si vous souhaitez répliquer l'affichage des données sous forme de tableau d'entrée pivoté, vous pouvez créer un tableau croisé dynamique équivalent dans une feuille.

Procédez comme suit :

1. Copiez et collez le tableau que vous venez de créer dans la feuille.
2. Faites glisser l'objet graphique **Tableau croisé dynamique** par-dessus la copie de tableau que vous venez de créer. Sélectionnez **Convertir**.
3. Cliquez sur ✔ **Édition terminée**.
4. Faites glisser le champ `Sales Person` de la partie supérieure de la colonne verticale vers la partie supérieure de la colonne horizontale.

Le tableau suivant montre les données sous leur forme de tableau initiale, telles qu'affichées dans Qlik Sense :

Tableau de résultats d'origine, comme affiché dans Qlik Sense

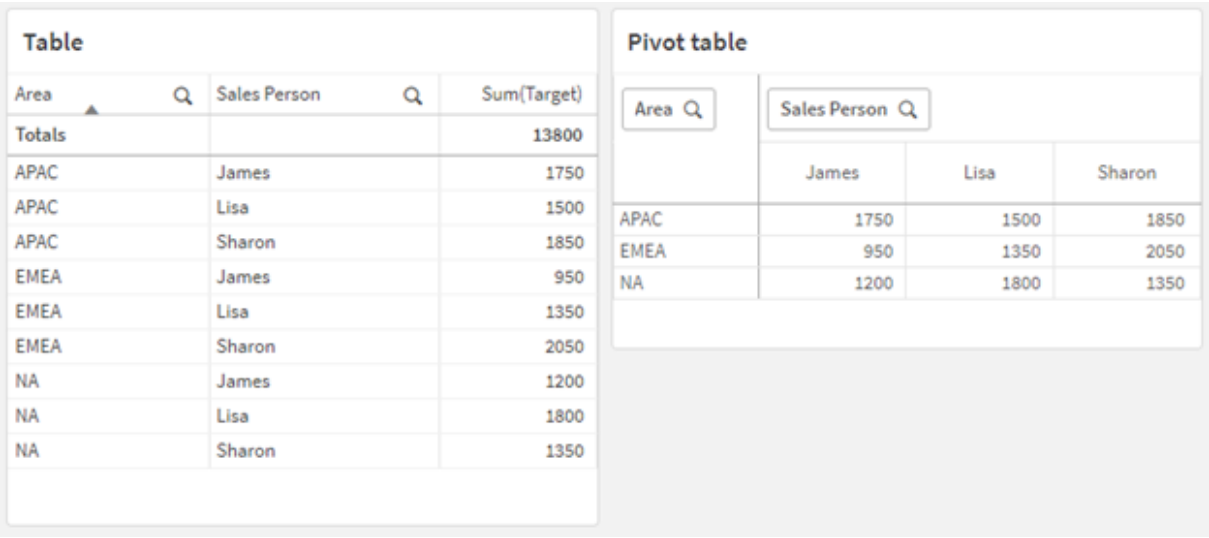
Area	Sales Person	=Sum(Target)
Totals	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Le tableau croisé dynamique équivalent ressemble à ce qui suit, la colonne du nom de chaque vendeur étant contenue dans la ligne plus grande pour `Sales Person` :

Tableau croisé dynamique équivalent avec le champ `Sales Person` pivoté à l'horizontale

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

Exemple de données affichées sous forme de tableau et un tableau croisé dynamique équivalent avec le champ `Sales Person` pivoté à l'horizontale



Table

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

Pivot table

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

Exemple 3 - Transformation de données sales et target pivotées en une structure de tableau verticale (avancé)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant les données sales et targets, organisées par secteur et mois de l'année. Cet ensemble de données est chargé dans une table appelée `SalesAndTargets`.
- Préfixe de chargement `crosstable`. Ceci est utilisé pour annuler le pivotement de la dimension `Month Year` dans un champ dédié, ainsi que pour transposer la matrice des montants sales et target dans un champ dédié appelé `Amount`.
- Conversion du champ `Month Year` du format texte en une date appropriée, à l'aide de la fonction de conversion de texte en date `date#`. Ce champ `Month Year` converti en date est de nouveau joint à la table `SalesAndTarget` via un préfixe de chargement `Join`.

Script de chargement

```
SalesAndTargets:
CROSSTABLE(MonthYearAsText,Amount,2)
LOAD
      *
INLINE [
Area   Type    Jan-22  Feb-22  Mar-22  Apr-22  May-22  Jun-22  Jul-22  Aug-22  Sep-22  Oct-22  Nov-22  Dec-22
APAC   Target  425     425     425     425     425     425     425     425     425     425     425     425
```

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

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

JOIN (SalesAndTargets)
LOAD * RESIDENT tmp;

DROP TABLE tmp;
DROP FIELD MonthYearAsText;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `Area`
- `Month Year`

Créez la mesure suivante, avec le libellé `Actual` :

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

Créez également la mesure suivante, avec le libellé `Target` :

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

Tableau de résultats (rogné)

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

Area	Month Year	Actual	Target
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Si vous souhaitez répliquer l'affichage des données sous forme de tableau d'entrée pivoté, vous pouvez créer un tableau croisé dynamique équivalent dans une feuille.

Procédez comme suit :

1. Copiez et collez le tableau que vous venez de créer dans la feuille.
2. Faites glisser l'objet graphique **Tableau croisé dynamique** par-dessus la copie de tableau que vous venez de créer. Sélectionnez **Convertir**.
3. Cliquez sur ✔ **Édition terminée**.
4. Faites glisser le champ `Month Year` de la partie supérieure de la colonne verticale vers la partie supérieure de la colonne horizontale.
5. Faites glisser l'élément `Values` de la partie supérieure de la colonne horizontale vers la partie supérieure de la colonne verticale.

Le tableau suivant montre les données sous leur forme de tableau initiale, telles qu'affichées dans Qlik Sense :

Tableau de résultats d'origine (rogné), comme affiché dans Qlik Sense

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

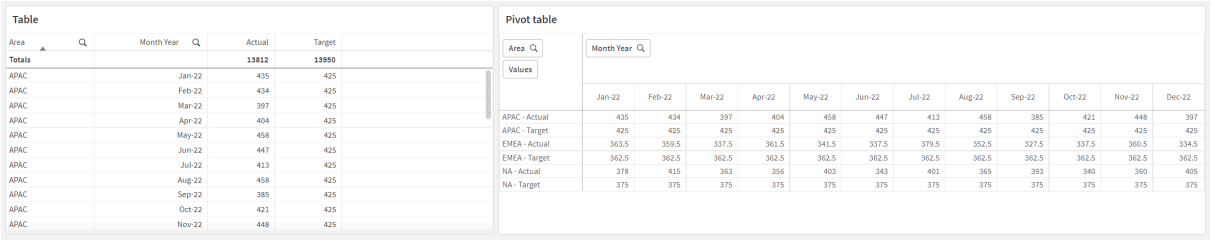
Area	Month Year	Actual	Target
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Le tableau croisé dynamique équivalent ressemble à ce qui suit, la colonne de chaque mois de l'année étant contenue dans la ligne plus grande pour `Month Year` :

Tableau croisé dynamique équivalent (rogné) avec le champ `Month Year` pivoté à l'horizontale

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

Exemple de données affichées sous forme de tableau et un tableau croisé dynamique équivalent avec le champ `Month Year` pivoté à l'horizontale



First

Le préfixe `First` associé à une instruction `LOAD` ou `SELECT` (SQL) sert à charger un nombre d'enregistrements maximal défini à partir d'une table de source de données. Un cas d'usage type pour l'utilisation du préfixe `First` peut se produire lorsque vous souhaitez récupérer un

petit sous-ensemble d'enregistrements à partir d'une étape de chargement de données importante et/ou lente. Dès que le nombre défini d'enregistrements "n" a été chargé, l'étape de chargement se termine prématurément et le reste de l'exécution du script se poursuit normalement.

Syntaxe :

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

Arguments

Argument	Description
n	Expression arbitraire qui aboutit à un entier indiquant le nombre maximal d'enregistrements à lire. n peut également figurer entre parenthèses : (n).
loadstatement | selectstatement	L'instruction load statement/select statement qui suit l'argument n définira la table spécifiée à charger avec le nombre maximal d'enregistrements défini.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction SET DateFormat de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
FIRST 10 LOAD * from abc.csv;	Cet exemple récupérera les dix premières lignes d'un fichier Excel.
FIRST (1) SQL SELECT * from Orders;	Cet exemple récupérera la première ligne sélectionnée de l'ensemble de données Orders.

Exemple – Chargement des cinq premières lignes

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates des deux premières semaines de 2020.
- Variable `First` qui demande à l'application de ne charger que les cinq premiers enregistrements.

Script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez `Date` comme champ et `sum(sales)` comme mesure.

Tableau de résultats

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

Le script charge uniquement les cinq premiers enregistrements de la table `Sales`.

Generic

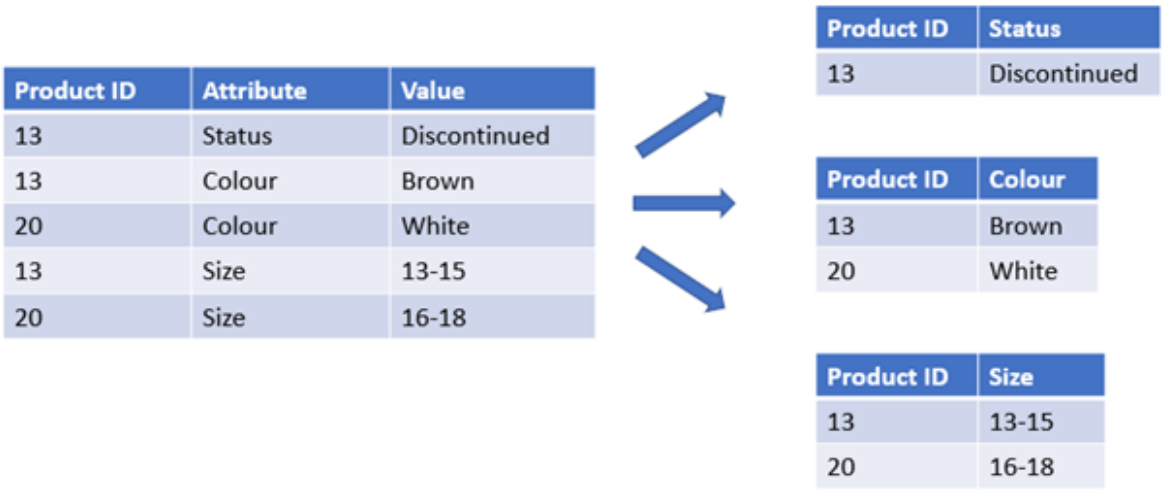
Le préfixe de chargement **Generic** permet la conversion des données modélisées entité-attribut-valeur (EAV) en une structure de table relationnelle normalisée standard. La modélisation EAV est également appelée « modélisation de données générique » ou « schéma ouvert ».

Exemple de données modélisées EAV et d'une table relationnelle dénormalisée équivalente

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

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

Exemple de données modélisées EAV et d'ensemble équivalent de tables relationnelles normalisées



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

Product ID	Status
13	Discontinued

Product ID	Colour
13	Brown
20	White

Product ID	Size
13	13-15
20	16-18

Bien qu'il soit techniquement possible de charger et d'analyser des données modélisées EAV dans Qlik, il est souvent plus facile de travailler avec une structure de données relationnelle traditionnelle équivalente.

Syntaxe :

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

Ces rubriques peuvent vous aider à utiliser cette fonction :

Rubriques connexes

Rubrique	Description
Crosstable (page 47)	Le préfixe de chargement `Crosstable` transforme les données orientées à l'horizontale en données orientées à la verticale. D'un point de vue purement fonctionnel, il effectue la transformation opposée à celle du préfixe de chargement `Generic`, même si les préfixes servent généralement des cas d'usage entièrement différents.
Bases de données génériques dans *Gestion des données*	Vous trouverez ici des modèles de données structurées EAV.

Exemple 1 – Transformation de données structurées EAV avec le préfixe de chargement Generic

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient un ensemble de données qui est chargé dans une table nommée `Transactions`. L'ensemble de données inclut un champ de date. La définition `MonthNames` par défaut est utilisée.

Script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `Color`.

Ajoutez cette mesure :

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

Vous pouvez maintenant inspecter le nombre de produits par couleur.

Tableau de résultats

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

Notez la forme du modèle de données, où chaque attribut a été divisé en une table distincte nommée en fonction de la balise de table cible d'origine `Product`. Chaque table a l'attribut comme suffixe. Par exemple, `Product.Color`. Les enregistrements de sortie Product Attribute obtenus sont associés par `Product ID`.

Représentation des résultats dans le visionneur de modèle de données

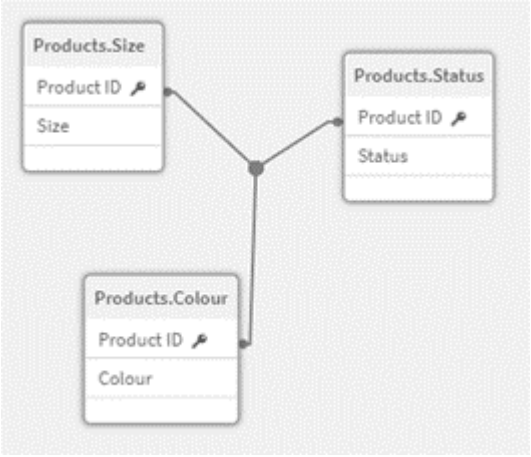


Tableau d'enregistrements obtenu : Products.Status

Product ID	Status
13	Discontinued
2	Discontinued

Tableau d'enregistrements obtenu : Products.Size

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

Tableau d'enregistrements obtenu : Products.Color

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

Exemple 2 – Analyse des données structurées EAV sans le préfixe de chargement Generic

Script de chargement et expression de graphique

Vue d'ensemble

Cet exemple montre comment analyser des données structurées EAV dans leur format d'origine.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient un ensemble de données chargé dans une table nommée `Products` dans une structure EAV.

Dans cet exemple, nous comptons toujours les produits par attribut de couleur. Afin d'analyser des données structurées de cette manière, vous devrez appliquer un filtrage au niveau de l'expression des produits portant la valeur Attribute `Color`.

De plus, les attributs individuels ne peuvent pas être sélectionnés sous forme de dimensions ou de champs, ce qui complique la création de visualisations efficaces.

Script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `Value`.

Créez la mesure suivante :

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

Vous pouvez maintenant inspecter le nombre de produits par couleur.

Tableau d'enregistrements obtenu : Products.Status

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

Exemple 3 – Dénormalisation des tableaux de sortie obtenus d'un chargement Generic (avancé)

Script de chargement et expression de graphique

Vue d'ensemble

Dans cet exemple, nous montrons comment la structure des données normalisée produite par le préfixe de chargement `Generic` peut être de nouveau dénormalisée dans un tableau de dimensions `Product` consolidé. Il s'agit d'une technique de modélisation avancée qui peut être utilisée dans le cadre du réglage fin des performances d'un modèle de données.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Script de chargement

```
Products:

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

RENAME TABLE Products.Color TO Products;

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

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

Résultats

Ouvrez le visionneur de modèle de données et notez la forme du modèle de données obtenu. Un seul tableau dénormalisé est présent. Il s'agit d'une combinaison des trois tableaux de sortie intermédiaires : `Products.Size`, `Products.Status` et `Products.Color`.

Modèle de données interne obtenu

Products
Product ID
Status
Color
Size

Tableau d'enregistrements obtenu : Products

Product ID	Status	Color	Size
13	Discontinued	Brown	13-15
20	-	White	16-18
2	Discontinued	White	-
5	-	Brown	-
44	-	Brown	-
45	-	Brown	16-18

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `Color`.

Ajoutez cette mesure :

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

Tableau de résultats

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

Hierarchy

Le préfixe **hierarchy** permet de transformer une table de hiérarchies parent-enfant en table utile dans un modèle de données Qlik Sense. Vous pouvez l'insérer devant une instruction **LOAD** ou **SELECT**. Il utilise le résultat de l'instruction de chargement comme entrée pour une transformation de table.

Le préfixe crée une table de nœuds étendus, qui contient normalement le même nombre d'enregistrements que la table d'entrée, à ceci près que chaque niveau de la hiérarchie est stocké en plus dans un champ distinct. Il est possible d'utiliser le champ du chemin d'accès dans une structure arborescente.

Syntaxe :

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

La table d'entrée doit être une table de nœuds adjacents. Les tables de nœuds adjacents sont des tables dans lesquelles chaque enregistrement correspond à un nœud et possède un champ contenant une référence au nœud parent. Dans une table de ce type, le nœud est stocké dans un seul enregistrement, mais il peut comporter plusieurs enfants. La table peut bien sûr comporter des champs supplémentaires décrivant les attributs des nœuds.

Le préfixe crée une table de nœuds étendus, qui contient normalement le même nombre d'enregistrements que la table d'entrée, à ceci près que chaque niveau de la hiérarchie est stocké en plus dans un champ distinct. Il est possible d'utiliser le champ du chemin d'accès dans une structure arborescente.

En général, la table d'entrée présente un enregistrement par nœud, tout comme la table de sortie. Cependant, certains nœuds présentent plusieurs parents, c'est-à-dire qu'un nœud est représenté par plusieurs enregistrements dans la table d'entrée. Dans ce cas, la table de sortie peut contenir plus d'enregistrements que la table d'entrée.

Tous les nœuds pour lesquels aucun ID de parent n'a été détecté dans la colonne de nœud d'ID (y compris les nœuds dont l'ID de parent est manquant) sont alors considérés comme des racines. En outre, seuls les nœuds connectés directement ou indirectement à un nœud racine sont chargés, évitant ainsi les références circulaires.

Des champs supplémentaires contenant le nom du nœud parent, le chemin d'accès et la profondeur du nœud peuvent être créés.

Arguments :

Arguments

Argument	Description
NodeID	Nom du champ contenant l'ID du nœud. Ce champ doit exister dans la table d'entrée.
ParentID	Nom du champ contenant l'ID du nœud parent. Ce champ doit exister dans la table d'entrée.
NodeName	Nom du champ contenant le nom du nœud. Ce champ doit exister dans la table d'entrée.
ParentName	Chaîne utilisée pour nommer le nouveau champ **ParentName**. Si cette chaîne est omise, le champ n'est pas créé.
ParentSource	Nom du champ contenant le nom du nœud utilisé pour créer le chemin d'accès au nœud. Paramètre facultatif. S'il est omis, **NodeName** est utilisé.
PathName	Chaîne utilisée pour nommer le nouveau champ **Path**, qui contient le chemin d'accès à la racine au nœud. Paramètre facultatif. Si cette chaîne est omise, le champ n'est pas créé.
PathDelimiter	Chaîne utilisée comme délimiteur dans le nouveau champ **Path**. Paramètre facultatif. S'il est omis, « / » est utilisé.
Depth	Chaîne utilisée pour nommer le nouveau champ **Depth**, qui contient la profondeur du nœud dans la hiérarchie. Paramètre facultatif. Si cette chaîne est omise, le champ n'est pas créé.

Exemple :

```
Hierarchy(NodeID, ParentID, NodeName, ParentName, NodeName, PathName, '\', Depth) LOAD *
inline [
NodeID, ParentID, NodeName
```

```
1, 4, London
2, 3, Munich
3, 5, Germany
4, 5, UK
5, , Europe
];
```

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

HierarchyBelongsTo

Ce préfixe permet de transformer une table de hiérarchies parent-enfant en table utile dans un modèle de données Qlik Sense. Vous pouvez l'insérer devant une instruction **LOAD** ou **SELECT**. Il utilise le résultat de l'instruction de chargement comme entrée pour une transformation de table.

Le préfixe crée une table contenant toutes les relations ancêtre-enfant de la hiérarchie. Les champs d'ancêtre peuvent alors être utilisés pour sélectionner des arborescences entières dans la hiérarchie. Dans la plupart des cas, la table de sortie contient plusieurs enregistrements par nœud.

Syntaxe :

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

La table d'entrée doit être une table de nœuds adjacents. Les tables de nœuds adjacents sont des tables dans lesquelles chaque enregistrement correspond à un nœud et possède un champ contenant une référence au nœud parent. Dans une table de ce type, le nœud est stocké dans un seul enregistrement, mais il peut comporter plusieurs enfants. La table peut bien sûr comporter des champs supplémentaires décrivant les attributs des nœuds.

Le préfixe crée une table contenant toutes les relations ancêtre-enfant de la hiérarchie. Les champs d'ancêtre peuvent alors être utilisés pour sélectionner des arborescences entières dans la hiérarchie. Dans la plupart des cas, la table de sortie contient plusieurs enregistrements par nœud.

Un champ supplémentaire contenant la différence de profondeur des nœuds peut être créé.

Arguments :

Arguments

Argument	Description
NodeID	Nom du champ contenant l'ID du nœud. Ce champ doit exister dans la table d'entrée.
ParentID	Nom du champ contenant l'ID du nœud parent. Ce champ doit exister dans la table d'entrée.
NodeName	Nom du champ contenant le nom du nœud. Ce champ doit exister dans la table d'entrée.
AncestorID	Chaîne utilisée pour nommer le nouveau champ d'ID d'ancêtre contenant l'ID du nœud ancêtre.
AncestorName	Chaîne utilisée pour nommer le nouveau champ d'ancêtre contenant le nom du nœud ancêtre.
DepthDiff	Chaîne utilisée pour nommer le nouveau champ **DepthDiff**, qui contient la profondeur du nœud dans la hiérarchie relative au nœud ancêtre. Paramètre facultatif. Si cette chaîne est omise, le champ n'est pas créé.

Exemple :

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

Results

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

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
4	4	UK	UK	0
4	5	UK	Europe	1
5	5	Europe	Europe	0

Inner

Les préfixes **join** et **keep** peuvent être précédés du préfixe **inner**. Utilisé avant **join**, il spécifie l'utilisation d'une jointure interne. De ce fait, la table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans les deux tables. Utilisé avant **keep**, il indique que les deux tables de données brutes doivent être réduites à leur intersection commune avant d'être stockées dans Qlik Sense.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Exemple

Script de chargement

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

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

Résultat

Table des résultats

Column1	Column2	Column3
A	B	C
1	aa	xx

Explication

Cet exemple démontre la sortie Inner Join, dans laquelle seules les valeurs présentes à la fois dans le premier tableau (gauche) et le deuxième tableau (droit) sont jointes.

IntervalMatch

Le préfixe **IntervalMatch** permet de créer une table faisant correspondre des valeurs numériques discrètes à un ou plusieurs intervalles numériques et, de manière facultative, faisant correspondre les valeurs d'une ou de plusieurs clés supplémentaires.

Syntaxe :

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

Le préfixe **IntervalMatch** doit être placé avant une instruction **LOAD** ou **SELECT** qui charge les intervalles. Le champ contenant les points de données discrètes (Time dans l'exemple ci-dessous) et les clés supplémentaires doit déjà avoir été chargé dans Qlik Sense avant l'instruction contenant le préfixe **IntervalMatch**. Le préfixe ne lit pas lui-même ce champ à partir de la table de la base de données. Le préfixe transforme la table d'intervalles et de clés chargée dans une table contenant une colonne supplémentaire : les points de données numériques discrètes. Il étend par ailleurs le nombre d'enregistrements de sorte que la nouvelle table contienne un enregistrement par combinaison possible de points de données discrètes, d'intervalle et de valeur du ou des champs clés.

Les intervalles peuvent se superposer et les valeurs discrètes sont alors liées à tous les intervalles correspondants.

Lorsque le préfixe IntervalMatch est étendu à l'aide de champs clés, il permet de créer une table faisant correspondre des valeurs numériques discrètes à un ou plusieurs intervalles numériques tout en correspondant simultanément aux valeurs d'une ou de plusieurs clés supplémentaires.

Afin d'éviter la non-prise en compte des limites d'intervalle non définies, il peut s'avérer nécessaire d'autoriser le mappage des valeurs NULL à d'autres champs qui constituent les limites inférieure et supérieure de l'intervalle. Cette opération peut être gérée par l'instruction **NullAsValue** ou par un test explicite qui remplace les valeurs NULL par une valeur numérique bien avant ou après les points de données numériques discrètes.

Arguments :

Arguments

Argument	Description
matchfield	Champ contenant les valeurs numériques discrètes à lier aux intervalles.
keyfield	Champs contenant les attributs supplémentaires auxquels les points doivent correspondre dans la transformation.

Argument	Description
loadstatement orselectstatement	Doit produire une table dont les deux premiers champs contiennent respectivement les limites inférieure et supérieure de chaque intervalle et, en cas d'utilisation de la correspondance de clés, le troisième champ et les suivants contiennent les champs clés figurant dans l'instruction **IntervalMatch**. Les intervalles sont toujours fermés, c'est-à-dire que les points de fin sont inclus dans l'intervalle. Les limites non numériques génèrent l'intervalle à ignorer (limites non définies).

Example 1:

Dans les deux tables ci-dessous, la première dresse la liste d'un certain nombre d'événements discrets tandis que la deuxième définit les heures de début et de fin pour la génération de différentes commandes. Au moyen du préfixe **IntervalMatch**, il est possible de connecter les deux tables de manière logique afin de rechercher, par exemple, les commandes ayant subi des perturbations et les commandes ayant été traitées par telle ou telle équipe.

```
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;
```

La table **OrderLog** contient à présent une colonne supplémentaire : *Time*. Le nombre d'enregistrements est également étendu.

Table with additional column

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

Example 2: (en utilisant keyfield)

Il s'agit du même exemple que ci-dessus, avec l'ajout de *ProductionLine* en tant que champ clé.

```
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;
```

Une zone table peut à présent être créée comme ci-dessous :

Tablebox example

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

Join

Le préfixe **join** permet de joindre la table chargée à une table nommée existante ou à la dernière table de données créée.

La jointure des données a pour effet d'ajouter au tableau cible un ensemble supplémentaire de champs ou d'attributs, à savoir ceux qui ne sont pas déjà présents dans le tableau cible. Tous les noms de champ communs entre l'ensemble de données source et le tableau cible sont utilisés pour déterminer comment associer les nouveaux enregistrements entrants. C'est ce qu'on appelle communément une « jointure naturelle ». Une opération de jointure Qlik peut réduire ou augmenter le nombre d'enregistrements de la table cible obtenue par rapport à la table de départ, suivant le caractère unique de l'association de jointure et le type de jointure utilisé.

Il existe quatre types de jointures :

Left join

Les jointures Left join (jointures à gauche) constituent le type de jointure le plus courant. Par exemple, si vous avez un ensemble de données de transaction et si vous souhaitez le combiner avec un ensemble de données de référence, vous utiliserez généralement une jointure `Left Join`. Vous devez d'abord charger la table de transactions, puis charger l'ensemble de données de référence tout en le joignant via un préfixe `Left Join` à la table de transactions déjà chargée. Une jointure `Left Join` conserverait toutes les transactions telles quelles et ajouterait les champs de données de référence supplémentaires lorsqu'une correspondance est trouvée.

Inner join

Lorsque vous avez deux ensembles de données pour lesquels vous ne vous souciez que des résultats où il existe une association correspondante, envisagez d'utiliser une jointure `Inner Join`. Cela éliminera tous les enregistrements des données source chargées et de la table cible si aucune correspondance n'est

trouvée. Par conséquent, cela peut aboutir à une table cible contenant moins d'enregistrements qu'avant l'opération de jointure.

Outer join

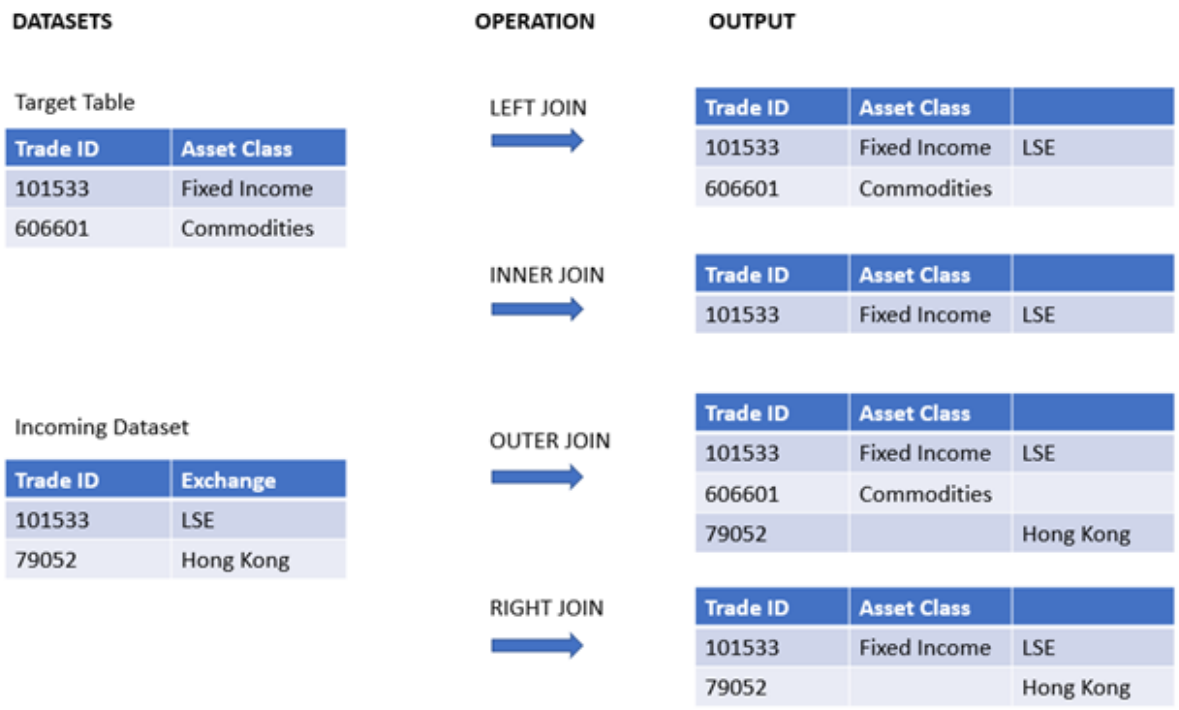
Lorsque vous devez conserver à la fois les enregistrements cible et tous les enregistrements entrants, utilisez une jointure `Outer Join`. Lorsqu'aucune correspondance n'est trouvée, chaque ensemble d'enregistrements est conservé, tandis que les champs du côté opposé de la jointure resteront vides (null). Les jointures externes (Outer join) ont généralement peu d'usages pratiques.

Right join

Ce type de jointure conserve tous les enregistrements sur le point d'être chargés, tout en réduisant les enregistrements de la table ciblée par la jointure aux seuls enregistrements pour lesquels il existe une correspondance d'association dans les enregistrements entrants. Il s'agit d'une jointure de type niche parfois utilisée pour réduire une table d'enregistrements déjà préchargée à un sous-ensemble requis.

Si le mot-clé de type est omis, `Inner Join` est le type de jointure par défaut.

Exemples d'ensembles de résultats de différents types d'opérations de jointure



S'il n'existe pas de noms de champ en commun entre la source et la cible d'une opération de jointure, la jointure se traduira par un produit cartésien de toutes les lignes - c'est ce qu'on appelle une « jointure croisée » (cross join).

Exemple d'ensemble de résultats d'une opération « cross join »

Syntaxe :

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

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Ces rubriques peuvent vous aider à utiliser cette fonction :

Rubriques connexes

Rubrique	Description
Combinaison de tables avec Join et Keep dans *Gestion des données*	Cette rubrique fournit des explications supplémentaires sur les concepts de « regroupement » (join) et de « conservation » (keep) des ensembles de données.
Keep (page 83)	Le préfixe de chargement `Keep` est similaire au préfixe `Join`, mais il ne combine pas les ensembles de données source et cible. Au lieu de cela, il rogne chaque ensemble de données en fonction du type d'opération adopté (inner, outer, left ou right).

Exemple 1 - Left join : Enrichissement d'une table cible à l'aide d'un ensemble de données de référence

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant des enregistrements de modification, chargé dans une table nommée `Changes`. Il inclut un champ clé Status ID.
- Deuxième ensemble de données représentant les statuts de modification, chargé et combiné avec les enregistrements de modification d'origine en le joignant avec un préfixe de chargement Left `Join`.

Cette jointure Left Join garantit que les enregistrements de modification restent intacts lors de l'ajout d'attributs de statut lorsqu'une correspondance dans les enregistrements de statut entrants est trouvée en fonction d'un Status ID commun.

Script de chargement

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

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

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

Résultats

Ouvrez le visionneur de modèle de données et notez la forme du modèle de données. Un seul tableau dénormalisé est présent. Il s'agit d'une combinaison de tous les enregistrements de modification d'origine, avec les attributs de statut correspondants joints à chaque enregistrement de modification.

Modèle de données interne obtenu

Changes
Change ID
Status ID
Scheduled Start Date
Scheduled End Date
Business Impact
Status
Sub Status

Si vous développez la fenêtre d'aperçu dans le visionneur de modèle de données, vous verrez une partie de cet ensemble de résultats complet organisé dans un tableau :

Aperçu du tableau Changes dans le visionneur de modèle de données

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10015	3	04/01/2022	15/02/2022	Low	Closed	Complete
10030	4	19/01/2022	23/02/2022	None	Closed	Cancelled
10031	3	20/01/2022	25/03/2022	Low	Closed	Complete
10040	1	29/01/2022	22/04/2022	None	Open	Not Started
10103	1	02/04/2022	29/05/2022	Medium	Open	Not Started
10116	1	15/04/2022	24/04/2022	None	Open	Not Started
10134	1	03/05/2022	08/07/2022	Low	Open	Not Started
10138	2	07/05/2022	03/08/2022	None	Open	Started
10185	2	23/06/2022	08/09/2022	None	Open	Started
10187	2	25/06/2022	24/08/2022	Low	Open	Started
10220	2	28/07/2022	06/09/2022	None	Open	Started

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10264	1	10/09/2022	17/10/2022	Medium	Open	Not Started
10323	1	08/11/2022	26/11/2022	High	Open	Not Started
10326	2	11/11/2022	05/12/2022	None	Open	Started
10334	2	19/11/2022	06/02/2023	Low	Open	Started

Revenez à l'éditeur de chargement de données. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `Status`.

Ajoutez cette mesure :

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

Vous pouvez maintenant inspecter le nombre de modifications (Changes) par statut (Status).

Tableau de résultats

Status	=Count([Change ID])
Open	12
Closed	3

Exemple 2 – Inner Join : Combinaison des enregistrements correspondants uniquement

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant des enregistrements de modification, chargé dans une table nommée `Changes`.
- Deuxième ensemble de données représentant les enregistrements de modification provenant du système source `JIRA`. Cet ensemble est chargé et combiné avec les enregistrements d'origine en le joignant avec un préfixe de chargement `Inner Join`.

Cette jointure `Inner Join` garantit que seuls les cinq enregistrements de modification trouvés dans les deux ensembles de données sont conservés.

Script de chargement

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

JIRA_changes:
Inner Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10187  JIRA
] (delimiter is '\t');
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- Source System
- Change ID
- Business Impact

Vous pouvez maintenant inspecter les cinq enregistrements obtenus.

Tableau de résultats

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	High
JIRA	10334	Low

Exemple 3 – Outer Join : Combinaison d'ensembles d'enregistrements en chevauchement

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant des enregistrements de modification, chargé dans une table nommée `Changes`.
- Deuxième ensemble de données représentant les enregistrements de modification provenant du système source `JIRA`. Cet ensemble est chargé et combiné avec les enregistrements d'origine en le joignant avec un préfixe de chargement `Outer Join`.

Cela garantit que tous les enregistrements de modification qui se chevauchent des deux ensembles de données sont conservés.

Script de chargement

```
// 8 Change records

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

```
// 6 Change records

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Vous pouvez maintenant inspecter les dix enregistrements obtenus.

Tableau de résultats

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	-
JIRA	10334	Low
JIRA	10597	-
-	10015	Low
-	10031	Low
-	10040	None
-	10138	None

Exemple 4 – Right Join : Réduction d'une table cible à un ensemble de données principal secondaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant des enregistrements de modification, chargé dans une table nommée `Changes`.
- Deuxième ensemble de données représentant les enregistrements de modification provenant du système source `Teamwork`. Cet ensemble est chargé et combiné avec les enregistrements d'origine en le joignant avec un préfixe de chargement `Right Join`.

Cela garantit que seuls les enregistrements de modification `Teamwork` sont conservés, sans perdre aucun enregistrement `Teamwork` si la table cible n'a pas de valeur `Change ID` correspondante.

Script de chargement

```
Changes:

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

Teamwork_changes:
Right Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
```

```
10040  Teamwork
10015  Teamwork
10103  Teamwork
10031  Teamwork
50231  Teamwork
] (delimiter is '\t');
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Vous pouvez maintenant inspecter les cinq enregistrements obtenus.

Tableau de résultats

Source System	Change ID	Business Impact
Teamwork	10015	Low
Teamwork	10031	Low
Teamwork	10040	None
Teamwork	10103	Medium
Teamwork	50231	-

Keep

Le préfixe **keep** est semblable au préfixe **join**. Tout comme le préfixe **join**, il compare la table chargée à une table nommée existante ou à la dernière table de données créée. Cependant, au lieu de joindre la table chargée à une table existante, il a pour effet de réduire une ou les deux tables avant qu'elles ne soient stockées dans Qlik Sense, en fonction de l'intersection des données des tables. La comparaison effectuée équivaut à une jonction naturelle entre tous les champs communs, c.-à-d. de la même manière que dans une jonction correspondante. Cependant, les deux tables ne sont pas jointes et sont conservées dans Qlik Sense comme deux tables nommées distinctes.

Syntaxe :

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

Le préfixe **keep** doit toujours être précédé par l'un des préfixes **inner**, **left** ou **right**.

Le préfixe **join** explicite du langage de script de Qlik Sense procède à une jointure complète des deux tables. Le résultat en est une seule table. De telles jointures produisent bien souvent des tables très volumineuses. L'une des principales caractéristiques de Qlik Sense est sa capacité à effectuer des associations entre plusieurs tables au lieu de les joindre, ce qui réduit considérablement l'utilisation de la

mémoire, augmente la vitesse de traitement et offre une grande souplesse. Il est donc généralement recommandé de ne pas utiliser de jointures explicites dans les scripts Qlik Sense. La fonctionnalité keep a été conçue pour réduire le nombre de cas d'utilisation de jointures explicites.

Arguments :

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Exemple :

```
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

Les préfixes **Join** et **Keep** peuvent être précédés du préfixe **left**.

Utilisé avant **join**, il spécifie l'utilisation d'une jointure gauche. La table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans la première table. Utilisé avant **keep**, il indique que la deuxième table de données brutes doit être réduite à son intersection commune avec la première table avant d'être stockée dans Qlik Sense.

Vous recherchez la fonction de chaîne du même nom ? Voir : Left (page 1447)

Syntaxe :

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

Arguments :

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Exemple

Script de chargement

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

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

Résultat

Table des résultats

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

Explication

Cet exemple démontre la sortie Left Join, dans laquelle seules les valeurs présentes dans le premier tableau (gauche) sont jointes.

Mappage

Le préfixe **mapping** permet de créer une table de mappage pouvant servir, par exemple, à remplacer des valeurs de champ et des noms de champ lors de l'exécution du script.

Syntaxe :

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

Vous pouvez insérer le préfixe **mapping** devant une instruction **LOAD** ou **SELECT**. Il stockera le résultat de l'instruction de chargement sous forme de table de mappage. Le préfixe mapping offre un moyen efficace de substituer des valeurs de champ lors de l'exécution du script, par ex. en remplaçant US, U.S. ou America par USA. Une table de mappage se compose de deux colonnes, la première contenant les valeurs de comparaison et la seconde contenant les valeurs de mappage voulues. Les tables de mappage sont stockées temporairement dans la mémoire et sont retirées automatiquement après l'exécution du script.

Le contenu de la table de mappage est accessible via, par exemple, l'instruction **Map … Using**, l'instruction **Rename Field**, la fonction **Applymap()** ou encore la fonction **Mapsubstring()**.

Exemple :

Dans cet exemple, nous chargeons une liste de représentants commerciaux accompagnés du code pays représentant leur pays de résidence. Nous utilisons une table mappant un code pays à un pays pour remplacer le code pays par le nom du pays. Seulement trois pays sont définis dans la table de mappage ;

les autres codes pays sont mappés à l'entrée 'Rest of the world' (Autre pays).

```
// 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';
```

La table résultante a l'aspect suivant :

Mapping table

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

Merge

Le préfixe **Merge** peut être ajouté à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier que la table chargée doit être fusionnée dans une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel.

Un cas d'utilisation type se produit, par exemple, lorsque vous chargez un journal des modifications et que vous souhaitez l'utiliser pour appliquer inserts, updates et deletes à une table existante.

Pour que le chargement partiel fonctionne correctement, vous devez ouvrir l'application avec des données avant le déclenchement du chargement partiel.

Effectuez un chargement partiel via le bouton **Charger**. Vous pouvez également utiliser Qlik Engine JSON API.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
only	Qualificateur facultatif indiquant que l'instruction doit être exécutée uniquement lors des chargements partiels. Lors des chargements normaux (non partiels), l'instruction est ignorée.
SequenceNoField	Nom du champ contenant un horodatage ou un numéro de séquence définissant l'ordre des opérations.
SequenceNoVar	Nom de la variable à laquelle est assignée la valeur maximale de SequenceNoField de la table en cours de fusion.
ListOfKeys	Liste séparée par des virgules de noms de champ spécifiant la clé primaire.
Operation	Le premier champ de l'instruction LOAD doit contenir l'opération sous forme de chaîne de texte : 'Insert', 'Update' ou 'Delete'. ‘i’, ‘u’ et ‘d’ sont également acceptés.

Fonctionnalité générale

Lors d'un chargement normal (non partiel), la construction **Merge LOAD** fonctionne comme une instruction **Load** normale, mais avec la fonctionnalité supplémentaire consistant à supprimer les enregistrements obsolètes plus anciens et les enregistrements indiqués comme devant être supprimés. Le premier champ de l'instruction **Load** doit contenir des informations sur l'opération : Insert, Update ou Delete.

Pour chaque enregistrement chargé, l'identificateur d'enregistrement est comparé aux enregistrements précédemment chargés et seul le dernier enregistrement (en fonction du numéro de séquence) est conservé. Si le dernier enregistrement est marqué à l'aide de Delete, aucun n'est conservé.

Table cible

La table à modifier est déterminée par l'ensemble de champs. Si une table avec le même ensemble de champs (sauf le premier champ ; l'opération) existe déjà, il s'agira de la table à modifier. Sinon, un préfixe **Concatenate** peut être utilisé pour spécifier la table. Si la table cible n'est pas déterminée, le résultat de la construction **Merge LOAD** est stocké dans une nouvelle table.

Si le préfixe Concatenate est utilisé, la table obtenue comprend un ensemble de champs correspondant à l'union de la table existante et de l'entrée de la fusion. Ainsi, la table cible peut recevoir plus de champs que le journal des modifications utilisé comme entrée de la fusion.

Un chargement partiel a le même résultat qu'un chargement complet. La différence réside dans le fait qu'un chargement partiel crée rarement une nouvelle table. À moins d'avoir utilisé la clause **Only**, il existe toujours une table cible avec le même ensemble de champs que l'exécution de script précédente.

Nombre de séquences

Si le journal des modifications chargé est un journal accumulé, à savoir, s'il contient des modifications déjà chargées, il est possible d'utiliser le paramètre SequenceNoVar dans une clause **Where** pour limiter le nombre de données d'entrée. La construction **Merge LOAD** peut alors s'appliquer uniquement aux enregistrements à charger dont le champ SequenceNoField est supérieur à SequenceNoVar. Une fois terminée, la construction **Merge LOAD** assigne une nouvelle valeur à SequenceNoVar avec la valeur maximale trouvée dans le champ SequenceNoField.

Opérations

La construction **Merge LOAD** peut contenir moins de champs que la table cible. Les différentes opérations traitent les champs manquants différemment :

Insert : Les champs manquants dans la construction **Merge LOAD**, mais existants dans la table cible, reçoivent une valeur NULL dans la table cible.

Delete : Les champs manquants n'affectent pas le résultat. Les enregistrements pertinents sont de toute façon supprimés.

Update : Les champs figurant dans **Merge LOAD** sont mis à jour dans la table cible. Les champs manquants ne sont pas modifiés. Cela signifie que les deux instructions suivantes ne sont pas identiques :

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

La première instruction met à jour les enregistrements répertoriés et remplace F2 par NULL. La deuxième instruction ne remplace pas F2, mais laisse les valeurs dans la table cible.

Exemples

Exemple 1 : Fusion simple avec une table spécifiée

Dans cet exemple, une table inline nommée Persons est chargée avec trois lignes. **Merge** modifie alors la table comme suit :

- Ajoute la ligne *Mary, 4*.
- Supprime la ligne *Steven, 3*.
- Assigne le numéro *5* à *Jake*.

La variable *LastChangeDate* est définie sur la valeur maximale de la colonne *ChangeDate* après l'exécution de la commande **Merge**.

Script de chargement

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
Set DateFormat='D/M/YYYY';
Persons:
load * inline [
Name, Number
Jake, 3
Jill, 2
Steven, 3
];

Merge (ChangeDate, LastChangeDate) on Name Concatenate(Persons)
LOAD * inline [
Operation, ChangeDate,   Name,     Number
Insert,    1/1/2021,     Mary,     4
Delete,    1/1/2021,     Steven,
Update,    2/1/2021,     Jake,     5
];
```

Résultat

Avant l'application de la construction **Merge Load**, la table obtenue apparaît comme suit :

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

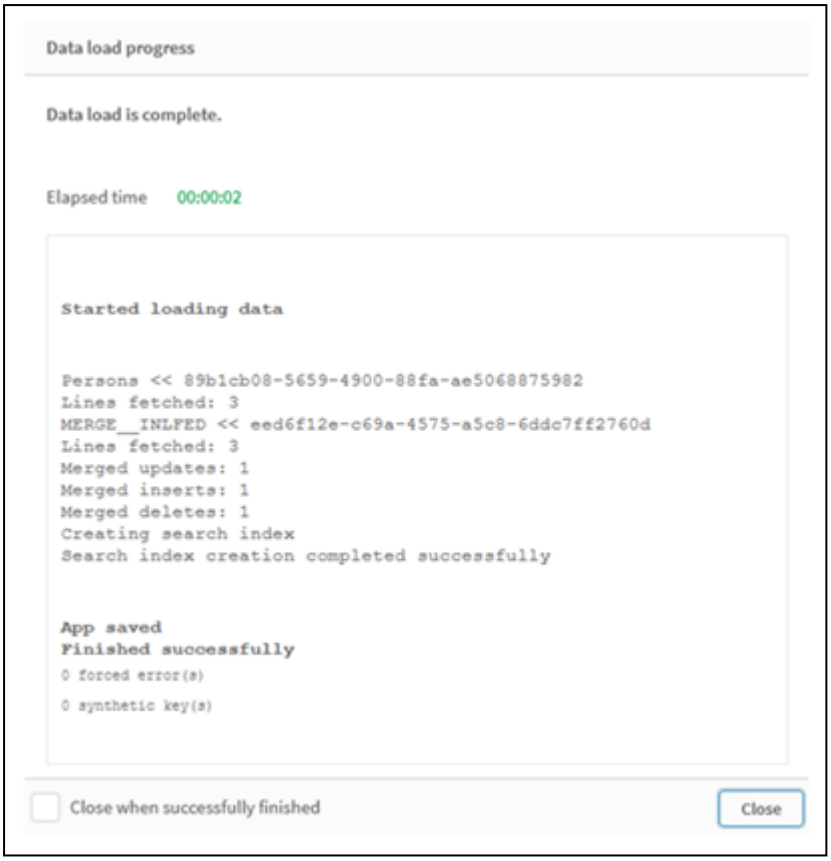
Après l'application de la construction **Merge Load**, la table apparaît comme suit :

Resulting table

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

Lors du chargement des données, la boîte de dialogue **Progression du chargement de données** indique les opérations effectuées :

Boîte de dialogue Progression du chargement de données



Exemple 2 : Script de chargement de données avec champs manquants

Dans cet exemple, les mêmes données que susmentionnées sont chargées, mais, à présent, avec un ID pour chaque personne.

Merge modifie la table comme suit :

- Ajoute la ligne *Mary, 4*.
- Supprime la ligne *Steven, 3*.
- Assigne le numéro *5* à *Jake*.
- Assigne le numéro *6* à *Jill*.

Script de chargement

Ici, nous utilisons deux instructions **Merge Load**, l'une pour 'Insert' et 'Delete' et la deuxième pour 'Update'.

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
Set DateFormat='D/M/YYYY';
Persons:
Load * Inline [
PersonID, Name, Number
1, Jake, 3
2, Jill, 2
3, Steven, 3
```

```
];

Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Name,     Number
Insert,    1/1/2021,     4,        Mary,     4
Delete,    1/1/2021,     3,        Steven,
];

Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Number
Update,    2/1/2021,     1,        5
Update,    3/1/2021,     2,        6
];
```

Résultat

Après l'application des instructions **Merge Load**, la table apparaît comme suit :

Resulting table

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

Notez que la deuxième instruction **Merge** n'inclut pas le champ **Name** et qu'en conséquence les noms n'ont pas été modifiés.

Exemple 3 : Script de chargement de données - chargement partiel via une clause Where avec ChangeDate

Dans l'exemple suivant, l'argument **Only** spécifie que la commande **Merge** est uniquement exécutée lors d'un chargement partiel. Les mises à jour sont filtrées en fonction de la valeur de la variable LastChangeDate précédemment capturée. Après l'application de la commande **Merge**, la variable LastChangeDate se voit assigner la valeur maximale de la colonne ChangeDate traitée lors de la fusion.

Script de chargement

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

NoConcatenate

Le préfixe **NoConcatenate** oblige deux tables chargées aux ensembles de champs identiques à être traitées comme deux tables internes distinctes. Sinon, elles seraient automatiquement concaténées.

Syntaxe :

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

Par défaut, si une table chargée contient un nombre identique de champs et des noms de champ identiques par rapport à une table précédemment chargée dans le script, Qlik Sense concatène automatiquement ces deux tables. Cela se produit, même si la deuxième table est nommée différemment.

Cependant, si le préfixe de script `NoConcatenate` est inclus avant l'instruction LOAD ou l'instruction SELECT de la deuxième table, ces deux tables sont chargées séparément.

Un cas d'usage type de `NoConcatenate` se produit si vous avez besoin de créer une copie temporaire d'une table pour effectuer des transformations temporaires sur cette copie, tout en conservant une copie des données originales. Grâce à `NoConcatenate`, vous pouvez effectuer cette copie sans la rajouter implicitement à la table source.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple de fonction

Exemple	Résultat
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Une table avec `A` et `B` comme mesures est chargée. Une deuxième table contenant les mêmes champs est chargée séparément via la variable `NoConcatenate`.

Exemple 1 – concaténation implicite

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, vous allez ajouter deux scripts de chargement dans l'ordre séquentiel.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données initial avec des dates et des montants, envoyé à une table appelée `Transactions`.

Premier script de chargement

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Premier tableau de résultats

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

Deuxième script de chargement

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un deuxième ensemble de données avec des champs identiques est envoyé à une table appelée `Sales`.

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

Résultats

Chargez les données et accédez à la table.

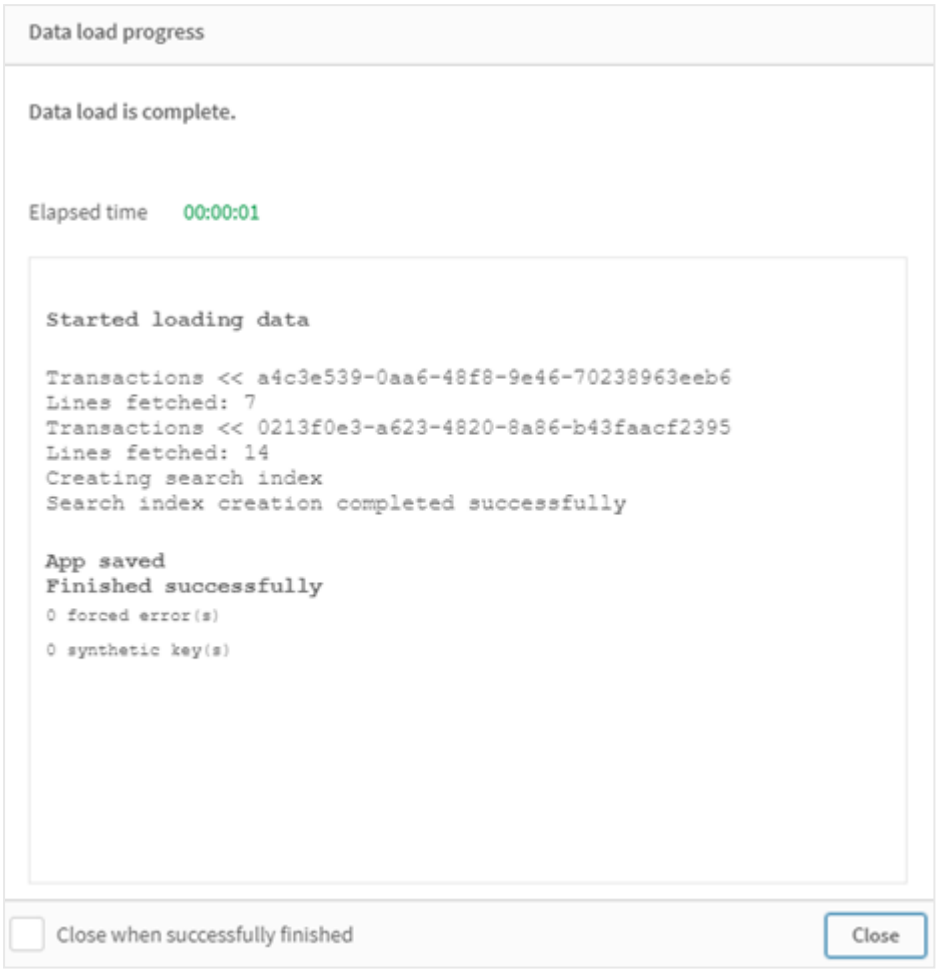
Deuxième tableau de résultats

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

Lors de l'exécution du script, la table `Sales` est implicitement concaténée à la table `Transactions` existante en raison du fait que les deux ensembles de données partage un nombre identique de champs portant des noms de champ identiques. Cela se produit, même si la balise de nom de la deuxième table tente de nommer l'ensemble de résultats `'Sales'`.

Vous pouvez voir que l'ensemble de données Sales est implicitement concaténé en consultant le journal **Progression du chargement de données**.

Journal Progression du chargement de données montrant la concaténation implicite des données Transactions.



Exemple 2 – scénario de cas d'usage

Script de chargement et résultats

Vue d'ensemble

Dans ce scénario de cas d'usage, vous avez :

- Un ensemble de données de transactions avec :
  - id
  - date
  - montant (en GBP)
- Une table de devises avec :
  - taux de conversion d'USD en GBP
- Un deuxième ensemble de données de transactions avec :
  - id

- date
- montant (en USD)

Vous allez charger cinq scripts dans l'ordre séquentiel.

- Le premier script de chargement contient un ensemble de données initial avec des dates et des montants en GBP, envoyé à une table appelée `Transactions`.
- Le deuxième script contient les éléments suivants :
  - Un deuxième ensemble de données avec des dates et des montants en USD, envoyé à une table appelée `Transactions_in_USD`.
  - Le préfixe `noconcatenate`, placé avant l'instruction LOAD de l'ensemble de données `Transactions_in_USD`, pour empêcher la concaténation implicite.
- Le troisième script de chargement contient le préfixe `join`, utilisé pour créer un taux de change de devises entre les devises GBP et USD dans la table `Transactions_in_USD`.
- Le quatrième script de chargement contient le préfixe `concatenate`, qui ajoutera la valeur `Transactions_in_USD` à la table `Transactions` initiale.
- Le cinquième script de chargement contient l'instruction `drop table`, qui supprimera la table `Transactions_in_USD` dont les données ont été concaténées à la table `Transactions`.

Premier script de chargement

```
Transactions:

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Résultats du premier script de chargement

id	date	amount
1	12/30/2018	23.56

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

Le tableau montre l'ensemble de données initial avec les montants en GDP.

Deuxième script de chargement

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

Résultats

Chargez les données et accédez à la table.

Résultats du deuxième script de chargement

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	164.27
9	01/03/2019	384.00

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

Vous verrez que le deuxième ensemble de données de la table `Transactions_in_USD` a été ajouté.

Troisième script de chargement

Ce script de chargement joint un taux de change de devises de la devise USD en devise GDP dans la table `Transactions_in_USD`.

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

Résultats

Chargez les données et accédez au visionneur de modèle de données. Sélectionnez la table `Transactions_in_USD` et vous verrez que chaque enregistrement existant comporte une valeur de champ 'rate' égale à 0,7.

Quatrième script de données

Via le chargement résident, ce script de chargement concatène la table `Transactions_in_USD` à la table `Transactions` après avoir converti les montants en USD.

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

Résultats

Chargez les données et accédez à la table. Vous verrez de nouvelles entrées avec des montants en GBP dans les lignes huit à quatorze.

Résultats du quatrième script de chargement

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

Cinquième script de chargement

Ce script de chargement abandonne les entrées en double du tableau de résultats du quatrième script de chargement, laissant uniquement les entrées avec des montants en GBP.

```
drop tables Transactions_in_USD;
```

Résultats

Chargez les données et accédez à la table.

Résultats du cinquième script de chargement

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

Après le chargement du cinquième script de chargement, le tableau de résultats montre l'ensemble des quatorze transactions qui existaient dans les deux ensembles de données de transactions ; en revanche, les montants des transactions 8 à 14 ont été convertis en GBP.

Si on supprime le préfixe `NoConcatenate` utilisé avant `Transactions_in_USD` dans le deuxième script de chargement, le script échoue avec l'erreur :“Table '`Transactions_in_USD`' not found” (Table introuvable). Cela est dû au fait que la table `Transactions_in_USD` aurait été automatiquement concaténée à la table `Transactions` originale.

Only

Le mot-clé de script **Only** s'utilise comme une fonction d'agrégation ou comme un élément de syntaxe dans les préfixes de chargement partiel **Add**, **Replace** et **Merge**.

Outer

Le préfixe explicite **Join** peut être précédé du préfixe **Outer** pour spécifier une jointure externe. Dans une jointure externe, toutes les combinaisons entre les deux tables sont générées. De ce fait, la table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans une ou les deux tables. Le mot-clé **Outer** est facultatif et correspond au type de jointure par défaut utilisé lorsqu'un préfixe de jointure n'est pas spécifié.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Exemple

Script de chargement

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

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

Tableau des résultats

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

Explication

Dans cet exemple, les deux tableaux, Table1 et Table2, sont fusionnés en un seul tableau intitulé Table1. Dans les cas comme celui-ci, le préfixe **outer** est souvent utilisé pour joindre plusieurs tableaux en un seul tableau afin d'effectuer des agrégations sur les valeurs d'un seul tableau.

Chargement partiel

Un chargement complet commence toujours par supprimer toutes les tables du modèle de données existant, puis exécute le script de chargement.

Un chargement partiel ne se comporte pas ainsi. Au lieu de cela, il conserve toutes les tables du modèle de données, puis exécute uniquement les instructions**Load** et **Select** précédées d'un préfixe **Add**, **Merge** ou **Replace**. Les autres tables de données ne sont pas affectées par la commande. L'argument **only** indique que l'instruction doit être exécutée uniquement lors des chargements partiels et qu'elle doit être ignorée lors des chargements complets. Le tableau suivant synthétise l'exécution des instructions pour les chargements partiels et complets.

Instruction	Chargement complet	Chargement partiel
Load ...	L'instruction sera exécutée	L'instruction ne sera pas exécutée
Add/Replace/Merge Load ...	L'instruction sera exécutée	L'instruction sera exécutée
Add/Replace/Merge Only Load ...	L'instruction ne sera pas exécutée	L'instruction sera exécutée

Les chargements partiels présentent plusieurs avantages par rapport aux chargements complets :

- Ils sont plus rapides, car seules les données récemment modifiées doivent être téléchargées. Avec des ensembles de données volumineux, la différence est importante.
- Ils consomment moins de mémoire, car ils chargent moins de données.
- Ils sont plus fiables, car les requêtes lancées auprès des données source sont plus rapides, ce qui réduit le risque de problèmes réseau.

Pour que le chargement partiel fonctionne correctement, vous devez ouvrir l'application avec des données avant le déclenchement du chargement partiel.

Effectuez un chargement partiel via le bouton **Charger**. Vous pouvez également utiliser Qlik Engine JSON API.

Limitation

Un chargement partiel peut supprimer des valeurs des données. Cependant, cela ne sera pas reflété dans la liste des valeurs distinctes, qui est une table maintenue en interne. Par conséquent, après un chargement partiel, la liste contiendra l'ensemble des valeurs distinctes qui existent dans le champ depuis le dernier chargement complet, qui peuvent être plus nombreuses que celles qui existent actuellement après le chargement partiel. Cela affecte le résultat des fonctions FieldValueCount() et FieldValue(). La fonction FieldValueCount() peut potentiellement renvoyer un nombre supérieur au nombre actuel de valeurs de champ.

Exemple

Exemple 1

Script de chargement

Ajoutez l'exemple de script à votre application et lancez un chargement partiel. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

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

Résultat

Resulting table

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

Explication

L'instruction est exécutée uniquement lors d'un chargement partiel. Si le préfixe "distinct" est omis, le nombre du champ **Num** augmentera à chaque chargement partiel suivant.

Exemple 2

Script de chargement

Ajoutez l'exemple de script à votre application. Lancez un chargement complet et affichez le résultat. Ensuite, lancez un chargement partiel et affichez le résultat. Pour afficher les résultats, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
T1:
Load recno() as ID, recno() as Value autogenerate 10;

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

Résultat

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4

ID	Value
5	5
6	6
7	7
8	8
9	9
10	10

Output table after partial reload

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

Explication

La première table est chargée lors d'un chargement complet et la deuxième table remplace simplement la première table lors d'un chargement partiel.

Replace

Le mot-clé de script **Replace** s'utilise comme fonction de script ou comme préfixe dans les rechargements partiels.

Replace

Le préfixe **Replace** peut être ajouté à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier que la table chargée doit remplacer une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Replace** peut également être utilisé dans une instruction **Map**.

Pour que le chargement partiel fonctionne correctement, vous devez ouvrir l'application avec des données avant le déclenchement du chargement partiel.

Effectuez un chargement partiel via le bouton **Charger**. Vous pouvez également utiliser Qlik Engine JSON API.

Syntaxe :

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

```
Replace [only] mapstatement
```

Lors d'un chargement normal (non partiel), la construction **Replace LOAD** fonctionne comme une instruction **LOAD** normale, mais précédée de **Drop Table**. Pour commencer, l'ancienne table est abandonnée, puis les enregistrements sont générés et stockés sous la forme d'une nouvelle table.

Si le préfixe **Concatenate** est utilisé ou s'il existe une table avec le même ensemble de champs, cette table sera celle à abandonner. Sinon, il n'existe aucune table à abandonner et la construction **Replace LOAD** est identique à une instruction **LOAD** normale.

Un chargement partiel fera la même chose. La seule différence réside dans le fait qu'il existe toujours une table de la précédente exécution de script à abandonner. La construction **Replace LOAD** commencera toujours par abandonner l'ancienne table avant d'en créer une nouvelle.

L'instruction **Replace Map...Using** permet également d'effectuer le mappage pendant une exécution de script partielle.

Arguments :

Arguments

Argument	Description
only	Qualificateur facultatif indiquant que l'instruction doit être exécutée uniquement lors des chargements partiels. Lors des chargements normaux (non partiels), elle doit être ignorée.

Exemples et résultats :

Exemple	Résultat
`Tab1: Replace LOAD * from File1.csv;`	Pendant les rechargements normaux et partiels, la table Qlik Sense Tab1 est d'abord retirée. De nouvelles données sont ensuite chargées à partir de File1.csv et stockées dans Tab1.

Exemple	Résultat
`Tab1: Replace only LOAD * from File1.csv;`	Pendant un rechargement normal, cette instruction est ignorée. Pendant un rechargement partiel, toute table Qlik Sense précédemment nommée Tab1 est d'abord retirée. De nouvelles données sont ensuite chargées à partir de File1.csv et stockées dans Tab1.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Pendant un rechargement normal, le fichier File1.csv est d'abord lu dans la table Tab1 de Qlik Sense, puis il est immédiatement retiré et remplacé par les nouvelles données chargées à partir du fichier File2.csv. Toutes les données provenant du fichier File1.csv sont perdues. Pendant un chargement partiel, la table Qlik Sense Tab1 toute entière est initialement abandonnée. Elle est ensuite remplacée par les nouvelles données chargées à partir du fichier File2.csv.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Pendant un rechargement normal, les données sont chargées à partir du fichier File1.csv et stockées dans la table Qlik Sense Tab1. Le fichier File2.csv est ignoré. Pendant un rechargement partiel, la table Qlik Sense Tab1 entière est d'abord retirée. Elle est ensuite remplacée par les nouvelles données chargées à partir du fichier File2.csv. Toutes les données provenant du fichier File1.csv sont perdues.

Right

Les préfixes **Join** et **Keep** peuvent être précédés du préfixe **right**.

Utilisé avant **join**, il spécifie l'utilisation d'une jointure droite. La table résultante contient uniquement des combinaisons de valeurs de champ provenant des tables de données brutes où les valeurs de champ de liaison sont représentées dans la deuxième table. Utilisé avant **keep**, il indique que la première table de données brutes doit être réduite à son intersection commune avec la deuxième table avant d'être stockée dans Qlik Sense.

Vous recherchez la fonction de chaîne du même nom ? Voir : Right (page 1455)

Syntaxe :

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

Arguments :

Arguments

Argument	Description
tablename	Table nommée à comparer à la table chargée.
loadstatementou selectstatement	Instruction **LOAD** ou **SELECT** de la table chargée.

Exemple

Script de chargement

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

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

Résultat

Table des résultats

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

Explication

Cet exemple démontre la sortie Right Join, dans laquelle seules les valeurs présentes dans le deuxième tableau (droit) sont jointes.

Sample

Le préfixe **sample** associé à une instruction **LOAD** ou **SELECT** permet de charger un échantillon aléatoire d'enregistrements à partir de la source de données.

Syntaxe :

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

L'expression qui est évaluée ne définit pas le pourcentage d'enregistrements de l'ensemble de données qui sera chargé dans l'application Qlik Sense, mais la probabilité du chargement de chaque enregistrement dans l'application. En d'autres termes, la spécification d'une valeur `p = 0.5` ne signifie pas que 50 % du nombre total d'enregistrements seront chargés, mais que chaque enregistrement aura 50 % de chances d'être chargé dans l'application Qlik Sense.

Arguments

Argument	Description
p	Expression arbitraire dont l'évaluation aboutit à un nombre supérieur à 0 et inférieur ou égal à 1. Ce nombre indique la probabilité qu'un enregistrement donné soit lu. Tous les enregistrements seront lus, mais seuls certains d'entre eux seront chargés dans Qlik Sense.

Cas d'utilisation

L'échantillon s'avère utile si vous souhaitez échantillonner des données provenant d'une table volumineuse pour comprendre la nature des données, leur distribution ou le contenu des champs. Comme il apporte un sous-ensemble de données, les chargements de données sont plus rapides, ce qui permet de

tester plus rapidement les scripts. Contrairement à la fonction `First`, la fonction `Sample` apporte des données de la table toute entière au lieu de se limiter aux premières lignes. Cela peut fournir une représentation plus exacte des données, dans certains cas.

Les exemples suivants montrent deux utilisations possibles du préfixe de script `Sample` :

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

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – échantillon d'une table inline

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, le script charge un échantillon d'ensemble de données provenant d'un ensemble de données contenant sept enregistrements dans une table appelée `Transactions` provenant d'une table inline.

Script de chargement

```
Transactions:
SAMPLE 0.3
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- amount

Ajoutez la mesure suivante :

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

Tableau de résultats

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

Dans l'itération du chargement de cet exemple, l'ensemble des sept enregistrements ont été lus, mais seuls quatre enregistrements ont été chargés dans la table de données. Toute réexécution du chargement peut produire un nombre différent et le chargement d'un ensemble d'enregistrements différents dans l'application.

Exemple 2 – échantillon provenant d'une table automatiquement générée

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, via `Autogenerate`, un ensemble de données de 100 enregistrements est créé avec les champs `date`, `id` et `amount`. Cependant, le préfixe `Sample` est utilisé, avec une valeur 0,1.

Script de chargement

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

Autogenerate(100);
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- amount

Ajoutez la mesure suivante :

Tableau de résultats

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

Dans l'itération du chargement de cet exemple, sept enregistrements ont été chargés à partir de l'ensemble de données créé. Encore une fois, toute réexécution du chargement peut produire un nombre différent et le chargement d'un ensemble d'enregistrements différents dans l'application.

Semantic

Le préfixe de chargement `semantic` crée un type de champ spécial qui peut être utilisé dans Qlik Sense pour connecter et gérer des données relationnelles telles que des structures arborescentes, des données structurées parent-enfant auto-référencées et/ou des données qui peuvent être décrites sous forme de graphique.

Notez que le chargement `semantic` peut fonctionner de la même manière que les préfixes *Hierarchy (page 65)* et *HierarchyBelongsTo (page 67)*. Les trois préfixes peuvent tous être utilisés comme blocs de construction dans des solutions frontales efficaces pour traverser des données relationnelles.

Syntaxe :

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

Un chargement semantic attend une entrée correspondant exactement à trois ou quatre champs de large avec une définition stricte de ce que chaque champ ordonné représente, comme indiqué dans le tableau ci-dessous :

Champs de chargement semantic

Nom de champ	Description du champ
1er champ :	Cette balise est une représentation du premier des deux objets entre lesquels il existe une relation.

Nom de champ	Description du champ
2e champ :	Cette balise sera utilisée pour décrire la relation "vers avant" entre le premier et le deuxième objets. Si le premier objet est un enfant et si le deuxième objet est un parent, vous pouvez créer un onglet de relation indiquant "parent" ou "parent of" comme si vous suiviez la relation d'enfant à parent.
3e champ :	Cette balise est une représentation du deuxième des deux objets entre lesquels il existe une relation.
4e champ :	Ce champ est facultatif. Cette balise décrit la relation "vers l'arrière" ou "inverse" entre le premier et le deuxième objets. Si le premier objet est un enfant et si le deuxième objet est un parent, un onglet de relation peut indiquer "child" ou "child of" comme si vous suiviez la relation de parent à enfant. Si vous n'ajoutez pas de quatrième champ, la balise du deuxième champ sera utilisée pour décrire la relation dans les deux sens. Dans ce cas, un symbole de flèche est automatiquement ajouté dans le cadre de la balise.

Le code suivant est un exemple du préfixe `semantic`.

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

Il est autorisé et pratique courante de libeller le troisième champ de la même manière que le premier champ. Cela crée une recherche d'auto-référencement qui vous permet de suivre le ou les objets jusqu'au ou aux objets associés, étape par étape. Si le troisième champ ne porte pas le même nom, le résultat final sera une simple recherche d'un ou de plusieurs objets vers son ou ses voisins relationnels directs distants d'une seule étape, ce qui présente peu d'utilité dans la pratique.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Fonctions associées

Fonctions	Interaction
Hierarchy (page 65)	Le préfixe de chargement Hierarchy est utilisé pour diviser et organiser les nœuds dans des structures de données parent-enfant et dans d'autres structures de données de type graphique et les transformer en tables.
HierarchyBelongsTo (page 67)	Le préfixe de chargement HierarchyBelongsTo est utilisé pour localiser et organiser les ancêtres de structures de données parent-enfant et d'autres structures de données de type graphique et les transformer en tables.

Exemple - Création d'un champ spécial pour connecter des relations à l'aide du préfixe semantic

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données représentant des enregistrements de relation géographique, chargé dans une table nommée `GeographyTree`.
    - Chaque entrée a un ID au début de la ligne et un ParentID à la fin de la ligne.
- Préfixe `semantic` qui ajoutera un champ de comportement spécial intitulé `Relation`.

Script de chargement

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

INLINE [
ID,Geography,ParentID
1,World
2,Europe,1
3,Asia,1
4,North America,1
5,South America,1
6,UK,2
7,Germany,2
8,Sweden,2
9,South Korea,3
```

```
10,North Korea,3
11,China,3
12,London,6
13,Birmingham,6
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `Id`
- `Geography`

Ensuite, créez un volet de filtre avec `Relation` comme dimension. Cliquez sur **Édition terminée**.

Tableau de résultats

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

Volet de filtre

Relation
Child
Parent

Cliquez sur **Europe** dans la dimension `Geography` du tableau, puis sur **Child** dans la dimension `Relation` du volet de filtre. Notez le résultat attendu dans le tableau :

Tableau de résultats montrant les relations "children" (enfants) de Europe

Id	Geography
6	UK
7	Germany
8	Sweden

Cliquez de nouveau sur **Child** pour afficher les lieux qui sont des "children" de UK, une étape plus bas.

Tableau de résultats montrant les relations "children" (enfants) de UK

Id	Geography
12	London
13	Birmingham

Unless

Utilisé comme préfixe ou comme suffixe, **unless** permet de créer une clause conditionnelle qui détermine si une instruction ou une clause exit doit être évaluée ou pas. Il peut être considéré comme une alternative plus compacte à l'instruction complète **if..end if**.

Syntaxe :

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

Les arguments **statement** ou **exitstatement** sont uniquement exécutés si l'argument **condition** est évalué comme False.

Vous pouvez utiliser le préfixe **unless** pour des instructions qui comportent déjà une ou plusieurs autres instructions, y compris des préfixes **when** ou **unless** supplémentaires.

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statement	Toute instruction de script Qlik Sense à l'exception des instructions de contrôle.
exitstatement	Toute clause **exit for**, **exit do** ou **exit sub**, ou toute instruction **exit script**.

Cas d'utilisation

L'instruction `Unless` renvoie un résultat booléen. En règle générale, ce type de fonction est utilisé comme condition lorsque l'utilisateur souhaite effectuer un chargement conditionnel ou exclure des parties du script.

Les lignes suivantes montrent trois exemples d'utilisation de la fonction `Unless` :

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

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – préfixe Unless

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Création de la variable A, d'une valeur 1.
- Ensemble de données chargé dans une table appelée Transactions, sauf si (unless) la variable A = 2.

Script de chargement

```
LET A = 1;

UNLESS A = 2

Transactions:
LOAD
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- amount

Tableau de résultats

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

Étant donné que la variable A a la valeur 1 au début du script, la condition suivant le préfixe `Unless` est évaluée, renvoyant un résultat `FALSE`. En conséquence, le script continue à exécuter l'instruction `Load`. Le tableau de résultats montre l'ensemble des enregistrements de la table `Transactions`.

Si la valeur de cette variable est définie sur 2, aucune donnée ne sera chargée dans le modèle de données.

Exemple 2 – suffixe Unless

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement commence par charger un ensemble de données initial dans une table appelée `Transactions`. Ensuite, le script se termine, sauf si (unless) il existe moins de dix enregistrements dans la table `Transactions`.

Si cette condition n'entraîne pas la fin du script, un autre ensemble de transactions est concaténé dans la table `Transactions` et ce processus est répété.

Script de chargement

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


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

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

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

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

```
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- amount

Tableau de résultats

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

Chacun des trois ensembles de données du script de chargement contient sept enregistrements.

Le premier ensemble de données (`id` de transaction de 1 à 7) est chargé dans l'application. La condition `Unless` évalue s'il existe moins de dix lignes dans la table `Transactions`. Le résultat est `TRUE`. Par conséquent, le deuxième ensemble de données (`id` de transaction 8 à 14) est chargé dans l'application. La deuxième condition `Unless` évalue s'il existe moins de dix enregistrements dans la table `Transactions`. Le résultat est `FALSE`, ce qui met fin au script.

Exemple 3 – plusieurs préfixes Unless

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Dans cet exemple, un ensemble de données contenant une transaction est créé dans une table appelée `Transactions`. Une boucle 'for' est alors déclenchée, dans laquelle deux instructions unless imbriquées sont évaluées :

1. Unless (sauf si) il existe plus de 100 enregistrements dans la table `Transactions`
2. Unless (sauf si) le nombre d'enregistrements dans la table `Transactions` est un multiple de 6

Si ces conditions sont évaluées sur `FALSE`, sept autres enregistrements sont générés et concaténés dans la table `Transactions` existante. Ce processus se répète jusqu'à ce qu'une des deux transactions renvoie une valeur `TRUE`.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`id`.

Tableau de résultats

id
0
1
2
3

id
4
5
+30 autres lignes

Les instructions unless imbriquées qui se produisent dans la boucle 'for' évaluent ce qui suit :

1. Existe-t-il plus de 100 lignes dans la table `Transactions` ?
2. Le nombre total d'enregistrements dans la table `Transactions` est-il un multiple de 6 ?

Chaque fois que les deux instructions unless renvoient une valeur `FALSE`, sept autres enregistrements sont générés et concaténés dans la table `Transactions` existante.

Ces instructions renvoient une valeur `FALSE` cinq fois, ce qui produit un total de 36 lignes de données dans la table `Transactions`.

Ensuite, la deuxième instruction `Unless` renvoie une valeur `TRUE`, et, par conséquent, l'instruction LOAD suivante ne sera plus exécutée.

When

Utilisé comme préfixe ou comme suffixe, **when** permet de créer une clause conditionnelle qui détermine si une instruction ou une clause exit doit être exécutée ou pas. Il peut être considéré comme une alternative plus compacte à l'instruction complète **if..end if**.

Syntaxe :

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

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

L'argument **statement** ou **exitstatement** est uniquement exécuté si la condition est évaluée sur TRUE.

Vous pouvez utiliser le préfixe `When` pour des instructions qui comportent déjà une ou plusieurs autres instructions, y compris des préfixes `When` ou `Unless` supplémentaires.

Cas d'utilisation

L'instruction `When` renvoie un résultat booléen. En règle générale, ce type de fonction est utilisé comme condition lorsque l'utilisateur souhaite charger ou exclure des parties du script.

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat TRUE ou FALSE
statement	Toute instruction de script Qlik Sense à l'exception des instructions de contrôle.
exitstatement	Toute clause **exit for**, **exit do** ou **exit sub**, ou toute instruction **exit script**.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`exit script when A=1;`	Lorsque l'instruction `A=1` est évaluée sur TRUE, le script s'arrête
`when A=1 LOAD * from myfile.csv;`	Lorsque l'instruction `A=1` est évaluée sur TRUE, le fichier `myfile.csv` est chargé.
`when A=1 unless B=2 drop table Tab1;`	Lorsque l'instruction `A=1` est évaluée sur TRUE, si `B=2` est évalué sur FALSE, la table `Tab1` est abandonnée.

Exemple 1 – préfixe When

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données avec des dates et des montants, envoyé à une table appelée '`Transactions`'.
- Instruction `Let`, qui déclare que la variable `A` est créée et que sa valeur est de `1`.
- Condition `When`, qui détermine que si `A` est égal à `1`, le script poursuit le chargement.

Script de chargement

```
LET A = 1;

WHEN A = 1

Transactions:
LOAD
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- amount

Tableau de résultats

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

Étant donné que la variable A a la valeur 1 au début du script, la condition suivant le préfixe When est évaluée et renvoie un résultat TRUE. Parce qu'elle renvoie un résultat TRUE, le script continue à exécuter l'instruction LOAD. Tous les enregistrements du tableau de résultats sont visibles.

Si la valeur de cette variable était définie sur une valeur différente de 1, aucune donnée ne serait chargée dans le modèle de données.

Exemple 2 – suffixe When

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Trois ensembles de données avec des dates et des montants, envoyés à une table appelée 'Transactions'.
    - Le premier ensemble de données contient les transactions 1 à 7.
    - Le deuxième ensemble de données contient les transactions 8 à 14.
    - Le troisième ensemble de données contient les transactions 15 à 21.
- Condition When qui détermine si la table 'Transactions' contient plus de dix lignes. Si une des instructions When est évaluée sur TRUE, le script de chargement s'arrête. Cette condition est placée à la fin de chacun des trois ensembles de données.

Script de chargement

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

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

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

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

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
```

```
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- amount

Tableau de résultats

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

Chacun des trois ensembles de données contient sept transactions. Le premier ensemble de données contient les transactions 1 à 7 et est chargé dans l'application. La condition `When` suivant cette instruction LOAD est évaluée sur FALSE, car la table '`Transactions`' contient moins de dix lignes. Le script de chargement passe à l'ensemble de données suivant.

Le deuxième ensemble de données contient les transactions 8 à 14 et est chargé dans l'application. La deuxième condition `When` est évaluée sur TRUE, car la table '`Transactions`' contient plus de dix lignes. Par conséquent, le script s'arrête.

Exemple 3 – plusieurs préfixes When

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant une seule transaction est créé dans une table appelée 'Transactions'.
- Une boucle For déclenchée contient deux conditions When imbriquées évaluées comme suit :
    1. Il existe moins de 100 enregistrements dans la table 'Transactions'.
    2. Le nombre d'enregistrements dans la table 'Transactions' n'est pas un multiple de 6.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- id

Le tableau de résultats montre uniquement les cinq premiers ID de transaction, mais le script de chargement crée 36 lignes, puis s'arrête une fois la condition When remplie.

Tableau de résultats

id
0
1
2

id
3
4
5
+30 autres lignes

Les conditions `When` imbriquées dans la boucle `For` évaluent les questions suivantes :

- Existe-t-il moins de 100 lignes dans la table '`Transactions`' ?
- Le nombre total d'enregistrements dans la table '`Transactions`' est-il différent d'un multiple de 6 ?

Chaque fois que les deux conditions `When` renvoient une valeur TRUE, sept autres enregistrements sont générés et concaténés dans la table '`Transactions`' existante.

Les conditions `When` renvoient une valeur TRUE à cinq reprises. À ce stade, la table '`Transactions`' contient un total de 36 lignes de données.

Lorsque 36 lignes de données sont créées dans la table '`Transactions`', la deuxième instruction `When` renvoie une valeur FALSE et, en conséquence, l'instruction LOAD suivante n'est plus exécutée.

2.5 Instructions normales de script

Les instructions normales servent généralement à manipuler des données d'une manière ou d'une autre. Ces instructions peuvent être écrites sur autant de lignes de script que nécessaire et doivent toujours se terminer par un point-virgule « ; ».

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

Vue d'ensemble des instructions normales de script

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Alias

L'instruction **alias** permet de définir un alias qui servira à renommer un champ chaque fois que celui-ci sera présent dans le script qui suit.

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

Autonumber

Cette instruction crée une valeur entière unique pour chaque valeur évaluée distincte d'un champ rencontrée au cours de l'exécution du script.

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

Binary

L'instruction **binary** permet de charger les données d'un autre document QlikView, y compris les données de Section Access.

```
Binary  [path] filename
```

comment

Permet d'afficher les commentaires (métadonnées) de champ à partir des bases de données et des tableurs. Les noms de champ absents de l'application sont ignorés. Si plusieurs occurrences d'un nom de champ sont détectées, la dernière valeur est utilisée.

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

comment table

Permet d'afficher les commentaires (métadonnées) de table à partir des bases de données ou des tableurs.

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

Connect

Cette fonctionnalité n'est pas disponible sous Qlik Sense SaaS.

L'instruction **CONNECT** permet de définir l'accès de Qlik Sense à une base de données générale via l'interface OLE DB/ODBC. Pour ODBC, la source de données doit d'abord être spécifiée à l'aide de l'administrateur 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

L'instruction **Declare** permet de créer des définitions de champ, qui précisent les relations entre des champs ou des fonctions. Un ensemble de définitions de champ peut servir à générer automatiquement des champs dérivés, que vous pouvez ensuite utiliser comme dimensions. Par exemple, vous pouvez créer une définition de calendrier, qui vous sert à générer les dimensions associées, telles que l'année, le mois, la semaine et le jour, à partir d'un champ de date.

```
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

L'instruction **Derive** permet de générer des champs dérivés à partir d'une définition de champ créée à l'aide de l'instruction **Declare**. Vous pouvez soit spécifier les champs de données pour lesquels les champs doivent être dérivés, soit dériver les champs de manière explicite ou implicite d'après les balises de champ.

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

Directory

L'instruction **Directory** définit le répertoire dans lequel le programme doit rechercher les fichiers de données dans les instructions **LOAD** ultérieures, jusqu'à ce qu'une nouvelle instruction **Directory** soit définie.

```
Directory [path]
```

Disconnect

L'instruction **Disconnect** met fin à la connexion ODBC/OLE DB/personnalisée active. Cette instruction est facultative.

```
Disconnect
```

drop field

Vous pouvez retirer un ou plusieurs champs Qlik Sense du modèle de données et, de ce fait, de la mémoire à tout moment au cours de l'exécution du script, au moyen d'une instruction **drop field**.

*Les formes **drop field** et **drop fields** sont toutes deux autorisées et leur effet est le même. Si aucune table n'est spécifiée, le champ est retiré de toutes les tables dans lesquelles il figure.*

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

drop table

Vous pouvez retirer une ou plusieurs tables internes Qlik Sense du modèle de données et, de ce fait, de la mémoire à tout moment au cours de l'exécution du script, au moyen d'une instruction **drop table**.

*Les formes **drop table** et **drop tables** sont toutes deux acceptées.*

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

Execute

L'instruction **Execute** permet d'exécuter d'autres programmes pendant que Qlik Sense est en train de charger des données. Elle s'utilise, par exemple, pour effectuer des conversions nécessaires.

```
Execute commandline
```

FlushLog

L'instruction **FlushLog** oblige Qlik Sense à consigner le contenu du tampon du script dans le fichier journal du script.

```
FlushLog
```

Force

L'instruction **force** oblige Qlik Sense à interpréter les noms et les valeurs de champ des instructions **LOAD** et **SELECT** ultérieures comme si elles étaient écrites en lettres majuscules uniquement, en lettres minuscules uniquement, commençant toujours par une majuscule ou telles qu'elles apparaissent (casse mixte). Cette instruction permet d'associer des valeurs de champ issues de tables élaborées selon différentes conventions.

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

LOAD

L'instruction **LOAD** charge des champs à partir d'un fichier, de données définies dans le script, d'une table déjà chargée, d'une page Web, du résultat d'une instruction **SELECT** ultérieure ou via la génération automatique de données. Il est également possible de charger des données à partir de connexions analytiques.

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

Let

L'instruction **let** complète l'instruction **set**, utilisée pour définir des variables de script. Contrairement à l'instruction **set**, l'instruction **let** évalue l'expression située à droite du signe '=' lors de l'exécution du script avant qu'elle soit attribuée à la variable.

```
Let variablename=expression
```

Loosen Table

Vous pouvez déclarer explicitement une ou plusieurs tables de données internes Qlik Sense comme déconnectées lors de l'exécution du script grâce à l'instruction **Loosen Table**. Lorsqu'une table est déconnectée, toutes les associations entre les valeurs de champ de la table sont supprimées. Il est

possible d'obtenir un effet similaire en chargeant chaque champ de la table déconnectée comme une table indépendante, sans lien. L'emploi de tables déconnectées peut s'avérer pratique lors des tests afin d'isoler temporairement différentes parties de la structure de données. Dans le visionneur de tables, une table déconnectée est signalée par des lignes en pointillé. L'utilisation d'une ou de plusieurs instructions **Loosen Table** dans le script permet à Qlik Sense d'ignorer toute déconnexion de tables définie avant l'exécution du script.

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

Map ... using

L'instruction **map ... using** permet de mapper une valeur de champ ou une expression donnée aux valeurs d'une table de mappage précise. La table de mappage est créée par l'instruction **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

L'instruction **NullAsNull** permet de désactiver la conversion des valeurs NULL en valeurs de chaîne définies précédemment au moyen d'une instruction **NullAsValue**.

```
NullAsNull *fieldlist
```

NullAsValue

L'instruction **NullAsValue** spécifie les champs pour lesquels il est nécessaire de convertir en valeur la valeur NULL rencontrée.

```
NullAsValue *fieldlist
```

Qualify

L'instruction **Qualify** permet d'activer la qualification des noms de champ, autrement dit les noms de champ se voient octroyer le nom de la table comme préfixe.

```
Qualify *fieldlist
```

Rem

L'instruction **rem** permet d'insérer des remarques ou des commentaires dans le script ou de désactiver temporairement des instructions de script sans pour autant les supprimer.

```
Rem string
```

Rename Field

Cette fonction de script permet de renommer un ou plusieurs champs Qlik Sense existants après leur chargement.

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

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

Rename Table

Cette fonction de script permet de renommer une ou plusieurs tables internes Qlik Sense existantes après leur chargement.

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

Section

L'instruction **section** permet de déterminer si les instructions **LOAD** et **SELECT** ultérieures doivent être considérées comme des données ou comme une définition des droits d'accès.

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

Select

La sélection de champs à partir d'une source de données ODBC ou d'un fournisseur OLE DB s'effectue au moyen d'instructions **SELECT** SQL standard. Cependant, l'acceptation des instructions **SELECT** dépend du pilote ODBC ou du fournisseur OLE DB utilisé.

```
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

L'instruction **set** permet de définir des variables de script. Ces variables peuvent servir à remplacer des chaînes, des chemins d'accès, des lecteurs, etc.

```
Set variablename=string
```

Sleep

L'instruction **sleep** interrompt l'exécution du script pendant la durée spécifiée.

```
Sleep n
```

SQL

L'instruction **SQL** vous permet d'envoyer une commande SQL arbitraire via une connexion ODBC ou OLE DB.

```
SQL sql_command
```

SQLColumns

L'instruction **sqlcolumns** renvoie un ensemble de champs qui décrit les colonnes d'une source de données ODBCou OLE DB à laquelle une instruction **connect** a été adressée.

```
SQLColumns
```

SQLTables

L'instruction **sqltables** renvoie un ensemble de champs qui décrit les tables d'une source de données ODBC ou OLE DB à laquelle une instruction **connect** a été adressée.

```
SQLTables
```

SQLTypes

L'instruction **sqltypes** renvoie un ensemble de champs décrivant les types d'une source de données ODBC ou OLE DB à laquelle une instruction **connect** a été adressée.

```
SQLTypes
```

Star

Vous pouvez utiliser l'instruction **star** pour définir la chaîne devant représenter l'ensemble des valeurs d'un champ dans la base de données. Elle affecte les instructions **LOAD** et **SELECT** ultérieures.

```
Star is [ string ]
```

Store

L'instruction **Store** crée un fichier QVD, CSV ou text.

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

Tag

Cette instruction de script permet d'assigner des balises à un ou plusieurs champs ou tables. Si vous tentez d'ajouter une balise à un champ ou une table qui ne figure pas dans l'application, l'opération est ignorée. Si des occurrences de nom de champ ou de balise conflictuelles sont détectées, la dernière valeur est utilisée.

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

Trace

L'instruction **trace** écrit une chaîne dans la fenêtre **Progression de l'exécution du script** et, le cas échéant, la consigne dans le fichier journal du script. Elle s'avère extrêmement pratique pour le débogage. L'utilisation d'expansions $ de variables calculées avant l'instruction **trace** vous permet de personnaliser le message.

```
Trace string
```

Unmap

L'instruction **Unmap** désactive le mappage de valeurs de champ spécifié par une instruction **Map … Using** précédente pour les champs chargés ultérieurement.

```
Unmap *fieldlist
```

Unqualify

L'instruction **Unqualify** permet de désactiver la qualification des noms de champ qui a été précédemment activée par l'instruction **Qualify**.

```
Unqualify *fieldlist
```

Untag

Cette instruction de script permet de supprimer des balises des champs ou des tables. Si vous tentez de supprimer une balise d'un champ ou d'une table qui ne figure pas dans l'application, l'opération est ignorée.

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

Alias

L'instruction **alias** permet de définir un alias qui servira à renommer un champ chaque fois que celui-ci sera présent dans le script qui suit.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
fieldname	Nom du champ figurant dans la source de données.
aliasname	Nom de l'alias à utiliser à la place.

Exemples et résultats :

Exemple	Résultat
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Les modifications de noms définies par cette instruction sont utilisées dans toutes les instructions **SELECT** et **LOAD** exécutées par la suite. Il est ensuite possible de définir un nouvel alias pour un nom de champ par une nouvelle instruction **alias** placée à n'importe quelle position ultérieure du script.

AutoNumber

Cette instruction crée une valeur entière unique pour chaque valeur évaluée distincte d'un champ rencontrée au cours de l'exécution du script.

Vous pouvez également faire appel à la fonction *autonumber (page 576)* au sein d'une instruction **LOAD**, mais cette méthode comporte des limitations en cas d'utilisation d'un chargement optimisé. Vous pouvez créer un chargement optimisé en commençant par charger les données à partir d'un fichier **QVD**, puis en convertissant les valeurs en clés de symbole à l'aide de l'instruction **AutoNumber**.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
*fieldlist	Liste de champs séparés par des virgules dont les valeurs doivent être remplacées par une valeur entière unique. Vous pouvez utiliser les caractères génériques ? et * dans les noms des champs afin d'inclure tous les champs dont les noms correspondent. De plus, le caractère * permet d'inclure tous les champs. Si vous utilisez des caractères génériques, vous devez placer les noms des champs entre guillemets.
namespace	**L'utilisation de** `namespace` est facultative. Cette option permet de créer un espace de noms, où les valeurs identiques des différents champs partagent la même clé. Si vous n'utilisez pas cette option, tous les champs seront dotés d'un index de clé distinct.

Limitations :

Si votre script comporte plusieurs instructions **LOAD**, vous devez placer l'instruction **AutoNumber** après la dernière instruction **LOAD**.

Exemple - script avec AutoNumber

Exemple de script

Dans cet exemple, les données commencent par être chargées sans l'instruction **AutoNumber**. L'instruction **AutoNumber** est ensuite ajoutée pour afficher l'effet.

Données utilisées dans l'exemple

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple de script ci-dessous. Pour le moment, laissez l'instruction **AutoNumber** commentée.

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

Création de visualisations

Créez deux visualisations de table dans une feuille Qlik Sense. Ajoutez **KeyToOtherTable**, **Region**, **Year**, **Month** et **Sales** comme dimensions à la première table. Ajoutez **KeyToOtherTable**, **Region**, **Year**, **Month** et **Budget** comme dimensions à la deuxième table.

Résultat

Table RegionSales

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

Table Budget

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Explication

L'exemple affiche un champ composite **KeyToOtherTable** qui lie les deux tables. L'instruction **AutoNumber** n'est pas utilisée. Notez la longueur des valeurs **KeyToOtherTable**.

Ajout de l'instruction AutoNumber

Décommentez l'instruction **AutoNumber** dans le script de chargement.

```
AutoNumber KeyToOtherTable;
```

Résultat

Table RegionSales

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

KeyToOtherTable	Region	Year	Month	Sales
4	South	2013	May	367
4	South	2014	June	645

Table Budget

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

Explication

Les valeurs du champ **KeyToOtherTable** ont été remplacées par des valeurs entières uniques et, en conséquence, la longueur des valeurs du champ a été réduite, ce qui préserve la mémoire. Les champs clés des deux tables sont affectés par l'instruction **AutoNumber** et les tables restent liées. L'exemple est court pour les besoins de la démonstration, mais il serait pertinent avec une table contenant un grand nombre de lignes.

Binary

L'instruction **binary** permet de charger les données d'une autre application Qlik Sense ou d'un document QlikView, y compris les données de l'accès de section. D'autres éléments de l'application ne sont pas inclus, tels que les feuilles, les récits, les visualisations, les éléments principaux ou les variables.

Le script n'admet qu'une seule instruction **binary**. L'instruction **binary** doit correspondre à la première instruction du script, devant même précéder les instructions SET habituellement situées au début du script.

Syntaxe :

```
binary [path] filename
```

Arguments :

Arguments

Argument	Description
path	Chemin d'accès au fichier, qui devrait être une référence à une connexion de données de type dossier. Obligatoire si le fichier ne se trouve pas dans le répertoire de travail de Qlik Sense. **Exemple :** *'lib://Table Files/'* En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data* • relatif à l'application contenant cette ligne de script. **Exemple :** *data*
filename	Nom du fichier, extension .qvw ou .qvf comprise.

Limitations :

Il n'est pas possible d'utiliser l'instruction **binary** pour charger des données à partir d'une application faisant partie du même déploiement Qlik Sense Enterprise en utilisant une référence à l'ID de l'application. Vous pouvez uniquement charger des données à partir d'un fichier *.qvf*.

Exemples

Chaîne	Description
`Binary lib://DataFolder/customer.qvw;`	Dans cet exemple, le fichier doit se trouver dans la connexion de données **Dossier**. Il peut s'agir, par exemple, d'un dossier créé par l'administrateur sur le serveur Qlik Sense. Cliquez sur **Créer une connexion** dans l'éditeur de chargement de données, puis sélectionnez **Dossier** sous **Emplacements de fichiers**.
`Binary customer.qvf;`	Dans cet exemple, le fichier doit se trouver dans le répertoire de travail de Qlik Sense.
`Binary c:\qv\customer.qvw;`	Cet exemple, qui utilise un chemin d'accès absolu, fonctionne uniquement en mode de script hérité.

Comment field

Permet d'afficher les commentaires (métadonnées) de champ à partir des bases de données et des tableurs. Les noms de champ absents de l'application sont ignorés. Si plusieurs occurrences d'un nom de champ sont détectées, la dernière valeur est utilisée.

Syntaxe :

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

La table de mappage utilisée doit comporter deux colonnes, la première contenant les noms des champs et la deuxième, les commentaires.

Arguments :

Arguments

Argument	Description
**fieldlist*	Liste des champs à commenter séparés par des virgules. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.
mapname	Nom d'une table de mappage déjà lue dans une instruction mapping **LOAD** ou mapping **SELECT**.
fieldname	Nom du champ à commenter.
comment	Commentaire à ajouter au champ.

Example 1:

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

Example 2:

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

Comment table

Permet d'afficher les commentaires (métadonnées) de table à partir des bases de données ou des tableurs.

Les noms de table absents de l'application sont ignorés. Si plusieurs occurrences d'un nom de table sont détectées, la dernière valeur est utilisée. Le mot-clé permet de lire des commentaires provenant d'une source de données.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
tablelist	(table{,table})
mapname	Nom d'une table de mappage déjà lue dans une instruction mapping **LOAD** ou mapping **SELECT**.
tablename	Nom de la table à commenter.
comment	Commentaire à ajouter à la table.

Example 1:

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

Example 2:

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

Connect

L'instruction **CONNECT** permet de définir l'accès de Qlik Sense à une base de données générale via l'interface OLE DB/ODBC. Pour ODBC, la source de données doit d'abord être spécifiée à l'aide de l'administrateur ODBC.

Cette fonctionnalité n'est pas disponible sous Qlik Sense SaaS.

Cette instruction prend uniquement en charge les connexions de données de type dossier en mode standard.

Syntaxe :

```
ODBC CONNECT TO connect-string
```

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

Arguments :

Arguments

Argument	Description

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	La chaîne de connexion se compose du nom de la source de données et d'une liste facultative d'un ou de plusieurs éléments de spécification de la connexion. Si le nom de la source de données contient des espaces ou si des éléments de spécification de la connexion sont précisés, la chaîne de connexion doit être mise entre guillemets.
	
	datasourcename doit correspondre à une source de données ODBC définie ou à une chaîne spécifiant un fournisseur OLE DB.
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	Les éléments de spécification de la connexion possibles peuvent différer d'une base de données à l'autre. Pour certaines bases de données, d'autres éléments que les éléments indiqués ci-dessus sont possibles. Pour une connexion OLE DB, certains éléments de spécification sont obligatoires au lieu d'être facultatifs.

connection	Nom d'une connexion de données stockée dans l'éditeur de chargement de données.

Si **ODBC** est placé avant **CONNECT**, l'interface ODBC est utilisée ; sinon, c'est OLE DB.

L'emploi de **LIB CONNECT TO** permet d'établir une connexion avec une base de données à l'aide d'une connexion de données stockée, créée dans l'éditeur de chargement de données.

Example 1:

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

La source de données définie dans cette instruction est utilisée par les instructions **Select (SQL)** ultérieures, jusqu'à ce qu'une nouvelle instruction **CONNECT** soit introduite.

Example 2:

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

Connect32

Cette instruction s'emploie de la même manière que l'instruction **CONNECT**, mais elle oblige un système 64 bits à utiliser un fournisseur ODBC/OLE DB 32 bits. Ne s'applique pas à custom connect.

Connect64

Cette instruction s'emploie de la même manière que l'instruction **CONNECT**, mais elle oblige le système à utiliser un fournisseur 64 bits. Ne s'applique pas à custom connect.

Declare

L'instruction **Declare** permet de créer des définitions de champ, qui précisent les relations entre des champs ou des fonctions. Un ensemble de définitions de champ peut servir à générer automatiquement des champs dérivés, que vous pouvez ensuite utiliser comme dimensions. Par exemple, vous pouvez créer une définition de calendrier, qui vous sert à générer les dimensions associées, telles que l'année, le mois, la semaine et le jour, à partir d'un champ de date.

L'instruction **Declare** vous permet de configurer une nouvelle définition de champ ou de créer une définition de champ basée sur une définition existante.

Configuration d'une nouvelle définition de champ

Syntaxe :

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

Arguments :

Argument	Description
definition_name	Nom de la définition de champ, se terminant par deux-points. *N'utilisez pas autoCalendar comme nom pour les définitions de champs, car ce nom est réservé aux modèles de calendrier générés automatiquement.* **Exemple :** `Calendar:`
tag_list	Liste de balises séparées par des virgules à appliquer à des champs dérivés de la définition de champ. L'application de balises est facultative, mais si vous n'appliquez pas de balises servant à spécifier l'ordre de tri, telles que $date, $numeric ou $text, le champ dérivé sera trié par défaut par ordre de chargement. **Exemple :** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`

Argument	Description
parameter_list	Liste de paramètres séparés par des virgules. Un paramètre se définit sous la forme `name=value` et se voit attribuer une valeur de départ, qu'il est possible de remplacer en cas de réutilisation d'une définition de champ. Facultatif. **Exemple :** `first_month_of_year = 1`
field_list	Liste de champs séparés par des virgules à générer lorsque la définition de champ est utilisée. Un champ se définit sous la forme `<expression> As field_name tagged tag`. Utilisez `$1` pour faire référence au champ de données à partir duquel les champs dérivés doivent être générés. **Exemple :** `Year($1) As Year tagged ('$numeric')`

Exemple :

```
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')
;
```

Le calendrier est à présent défini. Vous pouvez l'appliquer aux champs de date qui ont été chargés, dans ce cas OrderDate et ShippingDate, au moyen de la clause **Derive**.

Réutilisation d'une définition de champ existante

Syntaxe :

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

Arguments :

Argument	Description
definition_name	Nom de la définition de champ, se terminant par deux-points. **Exemple :** `MyCalendar:`
existing_definition	Définition de champ à réutiliser lors de la création de la nouvelle définition. La nouvelle définition de champ fonctionnera de la même façon que la définition initiale, à moins que vous n'utilisiez parameter_assignment pour modifier une valeur employée dans les expressions de champ. **Exemple :** `Using Calendar`
parameter_assignment	Liste d'affectations de paramètres séparées par des virgules. Une affectation de paramètre se définit sous la forme `name=value` et remplace la valeur de paramètre définie dans la définition du champ de base. Facultatif. **Exemple :** `first_month_of_year = 4`

Exemple :

Dans cet exemple, nous réutilisons la définition de calendrier créée au cours de l'exemple précédent. Dans ce cas, nous souhaitons utiliser un exercice fiscal qui commence en avril. Pour ce faire, nous affectons la valeur 4 au paramètre first_month_of_year, ce qui modifiera le champ DayNumberOfYear qui est défini.

Dans cet exemple, nous partons du principe que les échantillons de données et la définition des champs de l'exemple précédent sont réutilisés.

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

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

Lorsque vous avez rechargé le script de données, les champs générés sont disponibles dans l'éditeur de feuille, sous les noms OrderDate.MyCalendar.* et ShippingDate.MyCalendar.*.

Derive

L'instruction **Derive** permet de générer des champs dérivés à partir d'une définition de champ créée à l'aide de l'instruction **Declare**. Vous pouvez soit spécifier les champs de données pour lesquels les champs doivent être dérivés, soit dériver les champs de manière explicite ou implicite d'après les balises de champ.

Syntaxe :

```
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
```

Arguments :

Arguments

Argument	Description
definition	Nom de la définition de champ à utiliser lors de la dérivation de champs. **Exemple :** `Calendar`
field_list	Liste de champs de données séparés par des virgules à partir desquels les champs dérivés doivent être générés, en se basant sur la définition du champ. Les champs de données doivent correspondre à des champs que vous avez déjà chargés dans le script. **Exemple :** `OrderDate, ShippingDate`
tag_list	Liste de balises séparées par des virgules. Des champs dérivés seront générés pour tous les champs de données dotés des balises répertoriées. La liste de balises doit être placée entre parenthèses. **Exemple :** `('$date','$timestamp')`

Exemples :

- Dérivez les champs pour des champs de date précis.
  Dans ce cas, nous spécifions les champs OrderDate etShippingDate.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Dérivez les champs pour tous les champs dotés d'une balise spécifique.
  Dans ce cas, nous dérivons les champs basés sur le champ Calendar pour tous les champs dotés de la balise `$date`.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Dérivez les champs pour tous les champs dotés de la balise de définition de champ.
  Dans ce cas, nous dérivons les champs pour tous les champs de données disposant de la même balise que la définition de champ Calendar, soit `$date` dans cet exemple.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Directory

L'instruction **Directory** définit le répertoire dans lequel le programme doit rechercher les fichiers de données dans les instructions **LOAD** ultérieures, jusqu'à ce qu'une nouvelle instruction **Directory** soit définie.

Syntaxe :

```
Directory[path]
```

Si l'instruction **Directory** est émise sans argument **path** ou si elle est omise, Qlik Sense analyse le contenu du répertoire de travail Qlik Sense.

Arguments :

Arguments

Argument	Description
path	Texte pouvant être interprété comme chemin d'accès au fichier data. Le chemin d'accès correspond au chemin d'accès au fichier, sous l'une des formes suivantes : • absolu **Exemple :** *c:\data* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data* • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** *http://www.qlik.com*

Exemples :

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

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

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

Disconnect

L'instruction **Disconnect** met fin à la connexion ODBC/OLE DB/personnalisée active. Cette instruction est facultative.

Syntaxe :

```
Disconnect
```

Il est automatiquement mis fin à la connexion lorsqu'une nouvelle instruction **connect** est exécutée ou lorsque l'exécution du script est terminée.

Exemple :

```
Disconnect;
```

Drop

Le mot-clé de script **Drop** permet de retirer des tables ou des champs de la base de données.

Drop field

Vous pouvez retirer un ou plusieurs champs Qlik Sense du modèle de données et, de ce fait, de la mémoire à tout moment au cours de l'exécution du script, au moyen d'une instruction **drop field**.

*Les formes **drop field** et **drop fields** sont toutes deux autorisées et leur effet est le même. Si aucune table n'est spécifiée, le champ est retiré de toutes les tables dans lesquelles il figure.*

Syntaxe :

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

Exemples :

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

Drop table

Vous pouvez retirer une ou plusieurs tables internes Qlik Sense du modèle de données et, de ce fait, de la mémoire à tout moment au cours de l'exécution du script, au moyen d'une instruction **drop table**.

Syntaxe :

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

*Les formes **drop table** et **drop tables** sont toutes deux acceptées.*

Il en résulte que les éléments suivants seront perdus :

- Table(s) réelle(s).
- Tous les champs ne faisant pas partie des tables restantes.

- Valeurs de champ figurant dans les champs restants, provenant exclusivement de la ou des tables retirées.

Exemples et résultats :

Exemple	Résultat
`drop table Orders, Salesmen, T456a;`	Cette ligne a pour résultat le retrait de trois tables de la mémoire.

`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;`
	Dès que la table *Tab2* est créée, la table *Tab1* est retirée.

Drop table

Vous pouvez retirer une ou plusieurs tables internes Qlik Sense du modèle de données et, de ce fait, de la mémoire à tout moment au cours de l'exécution du script, au moyen d'une instruction **drop table**.

Syntaxe :

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

*Les formes **drop table** et **drop tables** sont toutes deux acceptées.*

Il en résulte que les éléments suivants seront perdus :

- Table(s) réelle(s).
- Tous les champs ne faisant pas partie des tables restantes.
- Valeurs de champ figurant dans les champs restants, provenant exclusivement de la ou des tables retirées.

Exemples et résultats :

Exemple	Résultat
`drop table Orders, Salesmen, T456a;`	Cette ligne a pour résultat le retrait de trois tables de la mémoire.

Exemple	Résultat
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;	Dès que la table *Tab2* est créée, la table *Tab1* est retirée.

Execute

L'instruction **Execute** permet d'exécuter d'autres programmes pendant que Qlik Sense est en train de charger des données. Elle s'utilise, par exemple, pour effectuer des conversions nécessaires.

Cette fonctionnalité n'est pas disponible sous Qlik Sense SaaS.

Cette instruction n'est pas prise en charge en mode standard.

Syntaxe :

```
execute commandline
```

Arguments :

Arguments

Argument	Description
commandline	Texte pouvant être interprété par le système d'exploitation comme une ligne de commande. Vous pouvez faire référence à un chemin d'accès absolu ou à un chemin d'accès au dossier lib://.

Si vous souhaitez utiliser **Execute** les conditions suivantes doivent être remplies :

- Le mode hérité doit être exécuté (condition valable pour Qlik Sense et Qlik Sense Desktop).
- Vous devez définir OverrideScriptSecurity sur 1 dans le fichier *Settings.ini* (condition valable pour Qlik Sense).
  Le fichier *Settings.ini* se trouve dans le dossier *C:\ProgramData\Qlik\Sense\Engine* et est généralement vide.

*Si vous définissez OverrideScriptSecurity de manière à activer **Execute**, tous les utilisateurs peuvent exécuter des fichiers sur le serveur. Par exemple, un utilisateur peut joindre un fichier exécutable à une application, puis exécuter le fichier dans le script de chargement de données.*

Procédez comme suit :

1. Effectuez une copie du fichier *Settings.ini* et ouvrez-la dans un éditeur de texte.
2. Vérifiez que le fichier inclut *[Settings 7]* sur la première ligne.
3. Insérez une nouvelle ligne et saisissez *OverrideScriptSecurity=1*.
4. Insérez une ligne vide à la fin du fichier.
5. Enregistrez le fichier.
6. Remplacez le fichier *Settings.ini* existant par le fichier que vous venez de modifier.
7. Redémarrez Qlik Sense Engine Service (QES).

Si Qlik Sense est exécuté en tant que service, certaines commandes peuvent se comporter de manière inattendue.

Exemple :

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

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

Field/Fields

Les mots-clés **Field** et **Fields** s'utilisent dans les instructions **Declare**, **Derive**, **Drop**, **Comment**, **Rename** et **Tag/Untag**.

FlushLog

L'instruction **FlushLog** oblige Qlik Sense à consigner le contenu du tampon du script dans le fichier journal du script.

Syntaxe :

```
FlushLog
```

Le contenu du tampon est consigné dans le fichier journal. Cette commande peut s'avérer utile à des fins de débogage, car vous disposez ainsi de données qui, sinon, auraient pu été perdues dans une exécution de script ayant échoué.

Exemple :

```
FlushLog;
```

Force

L'instruction **force** oblige Qlik Sense à interpréter les noms et les valeurs de champ des instructions **LOAD** et **SELECT** ultérieures comme si elles étaient écrites en lettres majuscules uniquement, en lettres minuscules uniquement, commençant toujours par une majuscule ou telles qu'elles apparaissent (casse mixte). Cette instruction permet d'associer des valeurs de champ issues de tables élaborées selon différentes conventions.

Syntaxe :

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

À défaut d'indication, c'est la casse mixte qui est utilisée. L'instruction force est valide jusqu'à ce qu'une nouvelle instruction force soit créée.

L'instruction **force** n'a pas d'effet sur Section Access : aucune des valeurs de champ chargées n'est sensible à la casse.

Exemples et résultats

Exemple	Résultat

Cet exemple illustre la manière de forcer la mise en majuscules de la première lettre.
 	
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La table **Capitalization** contient les valeurs suivantes :
	
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Toutes les valeurs commencent par une majuscule.

Cet exemple illustre la manière de forcer la mise en majuscules de toutes les lettres.
 	
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La table **CaseUpper**contient les valeurs suivantes :
	
	`AB`
	`CD`
	`EF`
	`GH`
	Toutes les valeurs sont en majuscules.

Exemple	Résultat

Cet exemple illustre la manière de forcer la mise en minuscules de toutes les lettres.
 	
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La table **CaseLower**contient les valeurs suivantes :
	
	`ab`
	`cd`
	`ef`
	`gh`
	Toutes les valeurs sont en minuscules.

Cet exemple illustre la manière de forcer l'emploi d'une casse mixte.
 	
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La table **CaseMixed**contient les valeurs suivantes :
	
	`ab`
	`Cd`
	`eF`
	`GH`
	Toutes les valeurs sont affichées telles qu'elles figurent dans le script.

Voir aussi :

From

Le mot-clé de script **From** s'utilise dans les instructions **Load** pour faire référence à un fichier et dans les instructions **Select** pour faire référence à une vue ou une table de base de données.

Load

L'instruction **LOAD** charge des champs à partir d'un fichier, de données définies dans le script, d'une table déjà chargée, d'une page Web, du résultat d'une instruction **SELECT** ultérieure ou via la génération automatique de données. Il est également possible de charger des données à partir de connexions analytiques.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
distinct	Vous pouvez utiliser **distinct** en tant que prédicat si vous ne souhaitez charger que des enregistrements uniques. En cas d'enregistrements en double, seule la première instance sera chargée. Si vous utilisez des instructions de chargement antérieures, vous devez placer **distinct** dans la première instruction LOAD, car **distinct** n'affecte que la table de destination.

Argument	Description
fieldlist	*fieldlist ::= (* | field {, * | field })* Liste des champs à charger. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs de la table. *field ::= (fieldref | expression)* [**as** *aliasname]* La définition du champ doit toujours contenir un littéral, une référence à un champ existant ou une expression. *fieldref ::= (fieldname* |*@fieldnumber* |*@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) *fieldname* est un texte identique à un nom de champ dans la table. Notez que le nom du champ doit être mis entre guillemets doubles droits ou entre crochets s'il contient des espaces, par exemple. Les noms des champs ne sont pas toujours disponibles de manière explicite. Une notation différente est alors utilisée : *@fieldnumber* représente le numéro du champ dans un fichier de table délimité. Il doit s'agir d'un entier positif précédé d'un arobase (@). La numérotation est toujours effectuée de 1 jusqu'au nombre total de champs. *@startpos*:*endpos* représente les positions de début et de fin d'un champ dans un fichier contenant des enregistrements de longueur fixe. Ces positions doivent être toutes deux des entiers positifs. Les deux nombres doivent être précédés d'un arobase (@) et séparés par deux-points. La numérotation est toujours effectuée de 1 jusqu'au nombre total de positions. Dans le dernier champ, **n** est utilisé comme position de fin. • Si *@startpos*:*endpos* est immédiatement suivi des caractères **I** ou **U**, les octets lus seront interprétés comme un entier binaire signé (**I**) ou non signé (**U**) (selon l'ordre des octets d'Intel). Le nombre de positions lues doit être égal à 1, 2 ou 4. • Si *@startpos*:*endpos* est immédiatement suivi du caractère **R**, les octets lus seront interprétés comme un nombre réel binaire (en virgule flottante de 32 bits ou 64 bits IEEE). Le nombre de positions lues doit être égal à 4 ou 8. • Si *@startpos*:*endpos* est immédiatement suivi du caractère **B**, les octets lus seront interprétés comme des nombres BCD (Binary Coded Decimal) selon la norme COMP-3. Vous pouvez spécifier n'importe quel nombre d'octets. *expression* peut correspondre à une fonction numérique ou une fonction de chaîne basée sur un ou plusieurs autres champs de la même table. Pour plus d'informations, voir la syntaxe des expressions. **as** est utilisé pour attribuer un nouveau nom au champ.

Argument	Description
from	**from** est utilisé si les données doivent être chargées à partir d'un fichier au moyen d'une connexion de données de type Dossier ou Fichier Web. *file ::= [path] filename* **Exemple :** ***'lib://Table Files/'*** Si le chemin d'accès est omis, Qlik Sense recherche le fichier dans le répertoire que lui indique l'instruction **Directory**. En l'absence d'instruction **Directory**, Qlik Sense effectue une recherche dans le répertoire de travail, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *Dans une installation de serveur Qlik Sense, le répertoire de travail est spécifié dans Qlik Sense Repository Service ; par défaut, il s'agit de C:\ProgramData\Qlik\Sense\Apps.* L'argument *filename* peut contenir les caractères génériques DOS standard (* et ?). Tous les fichiers correspondants sont alors chargés dans le répertoire indiqué. *format-spec ::= (fspec-item { , fspec-item })* La spécification du format se compose d'une liste de plusieurs éléments de spécification du format, mis entre parenthèses. **Mode de script hérité** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** ***c:\data*** • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** ***data*** • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** ***http://www.qlik.com***

Argument	Description
from_field	**from_field** est utilisé si les données doivent être chargées à partir d'un champ précédemment chargé. *fieldassource::=*(*tablename, fieldname*) Le champ correspond au nom des arguments *tablename* et *fieldname* précédemment chargés. *format-spec ::= (fspec-item {, fspec-item })* La spécification du format se compose d'une liste de plusieurs éléments de spécification du format, mis entre parenthèses.
inline	**inline** est utilisé si les données doivent être saisies dans le script au lieu d'être chargées à partir d'un fichier. *data ::= [text]* Les données saisies à l'aide d'une clause **inline** doivent être mises entre guillemets ou entre crochets. Le texte placé entre ces guillemets ou crochets est interprété de la même manière que le contenu d'un fichier. C'est pourquoi vous devez également insérer une nouvelle ligne dans la clause **inline**, là où vous en auriez inséré une dans un fichier texte. Pour ce faire, appuyez sur la touche Entrée lors de la saisie du script. Le nombre de colonnes est défini dans la première ligne. *format-spec ::= (fspec-item {, fspec-item })* La spécification du format se compose d'une liste de plusieurs éléments de spécification du format, mis entre parenthèses.
resident	**resident** est utilisé si les données doivent être chargées à partir d'une table précédemment chargée. *table label* est une étiquette précédant l'instruction **LOAD** ou **SELECT** ayant créé la table de départ. L'étiquette doit être saisie avec deux-points à la fin.
autogenerate	**autogenerate** est utilisé si les données doivent être générées automatiquement par Qlik Sense. *size ::= number* *Number* est un entier indiquant le nombre d'enregistrements à générer. La liste de champs ne doit pas contenir d'expressions nécessitant des données provenant d'une source de données externe ou d'une table chargée précédemment, à moins que vous ne fassiez référence à une valeur de champ unique dans une table chargée au préalable à l'aide de la fonction **Peek**.

Argument	Description
extension	Vous pouvez charger des données à partir de connexions analytiques. Vous devez utiliser la clause **extension** pour appeler une fonction définie dans le plug-in SSE (Server-Side Extension) ou pour évaluer un script. Il est possible d'envoyer une seule table au plug-in SSE ; une seule table de données est alors renvoyée. Si le plug-in ne précise pas les noms des champs renvoyés, les champs seront nommés Field1, Field2, et ainsi de suite. `Extension pluginname.functionname( tabledescription );` • Chargement de données à l'aide d'une fonction dans un plug-in SSE *tabledescription ::= (table { ,tablefield})* Si vous ne déclarez pas les champs de table, ils sont utilisés dans l'ordre de chargement. • Chargement de données via l'évaluation d'un script dans un plug-in SSE *tabledescription ::= (script, table { ,tablefield})* **Gestion des types de données dans la définition des champs de table** Les types de données sont automatiquement détectés dans les connexions analytiques. Si les données ne comportent aucune valeur numérique et comprennent au moins une chaîne de texte non NULLE, le champ est interprété comme du texte. Dans tous les autres cas, il est considéré comme de type numérique. Vous pouvez appliquer un type de données forcé en encadrant le nom d'un champ à l'aide de **String()** ou de **Mixed()**. • **String()** convertit de force le champ en texte. Si le champ est numérique, la partie texte de la valeur double est extraite et aucune conversion n'est effectuée. • **Mixed()** convertit de force le champ en valeur double. Il est impossible d'utiliser **String()** ou **Mixed()** en dehors des définitions de champ de table **extension**. De plus, vous ne pouvez pas utiliser d'autres fonctions Qlik Sense dans une définition de champ de table. **Informations complémentaires sur les connexions analytiques** Avant de pouvoir utiliser des connexions analytiques, vous devez les configurer.
where	**where** est une clause utilisée pour indiquer si un enregistrement doit être inclus ou pas dans la sélection. La sélection est incluse si l'expression *criterion* est définie sur True. *criterion* est une expression logique.

Argument	Description
while	**while** est une clause utilisée pour indiquer si un enregistrement doit être lu plusieurs fois. Le même enregistrement est lu tant que l'expression *criterion* est définie sur True. Pour être utile, une clause **while** doit généralement inclure la fonction **IterNo( )**. *criterion* est une expression logique.
group by	**group by** est une clause utilisée pour déterminer les champs sur lesquels les données doivent être agrégées (groupées). Les champs d'agrégation doivent être inclus d'une manière ou d'une autre dans les expressions chargées. Aucun autre champ que les champs d'agrégation ne peut être utilisé en dehors des fonctions d'agrégation dans les expressions chargées. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** est une clause utilisée pour trier les enregistrements d'une table résidente avant qu'ils ne soient traités par l'instruction **load**. La table résidente peut être triée par un ou plusieurs champs, par ordre croissant ou décroissant. Le tri est principalement effectué par valeur numérique, puis par valeur de paramètres régionaux de classement national. Cette clause peut uniquement être utilisée lorsque la source de données est une table résidente. Les champs de tri indiquent les champs selon lesquels la table résidente est triée. Le champ peut être spécifié par son nom ou par son numéro dans la table résidente (le premier champ est le numéro 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* correspond soit à *asc* pour un ordre croissant, soit à *desc* pour un ordre décroissant. Si aucun argument *sortorder* n'est spécifié, c'est *asc* qui est utilisé. *fieldname*, *path*, *filename* et *aliasname* sont des chaînes textuelles représentant ce que ces noms désignent. N'importe quel champ de la table source peut être utilisé comme *fieldname*. Toutefois, les champs créés via la clause as (*aliasname*) ne sont pas concernés et ne peuvent pas être utilisés dans la même instruction **load**.

Si aucune source de données n'est fournie par les clauses **from**, **inline**, **resident**, **from_field, extension** ou **autogenerate**, les données sont chargées à partir du résultat de l'instruction **SELECT** ou **LOAD** qui suit immédiatement. L'instruction qui suit ne doit pas comporter de préfixe.

Exemples :

Chargement de différents formats de fichier
Chargez un fichier de données délimité défini à l'aide des options par défaut :

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

Chargez un fichier de données délimité à partir d'une connexion de bibliothèque (DataFiles) :

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

Chargez tous les fichiers de données délimités à partir d'une connexion de bibliothèque (DataFiles) :

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

Chargez un fichier délimité, en spécifiant la virgule comme délimiteur et en incluant les étiquettes incorporées :

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

Chargez un fichier délimité, en spécifiant la tabulation comme délimiteur et en incluant les étiquettes incorporées :

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

Chargez un fichier dif avec en-têtes incorporés :

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

Chargez trois champs à partir d'un fichier d'enregistrements fixes sans en-têtes :

```
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);
```

Chargez un fichier QVX en spécifiant un chemin d'accès absolu :

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

Chargement de fichiers Web

Chargez un fichier à partir de l'URL par défaut définie dans la connexion de données de type Fichier Web :

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

Chargez un fichier à partir d'une URL spécifique et remplacez l'URL définie dans la connexion de données de type Fichier Web :

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

Chargez un fichier à partir d'une URL spécifique définie dans une variable à l'aide d'une expansion $:

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

Sélection de certains champs, modification de noms de champ et calcul de champs

Chargez uniquement trois champs spécifiques à partir d'un fichier délimité :

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

Renommez le premier champ A et le deuxième champ B lors du chargement du fichier sans étiquettes :

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

Chargez le nom (Name) sous la forme d'une concaténation du prénom, d'un espace et du nom de famille, soit FirstName, espace et LastName :

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

Chargez Quantity, Price et Value (le produit de Quantity et Price) :

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

Sélection de certains enregistrements
Chargez exclusivement des enregistrements uniques, les doublons étant ignorés :

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

Chargez uniquement des enregistrements où le champ Litres est doté d'une valeur supérieure à zéro :

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

Chargement de données ne se trouvant pas dans un fichier et de données générées automatiquement
Chargez une table comportant des données intégrées, deux champs nommés CatID et Category :

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

Chargez une table comportant des données intégrées, trois champs nommés UserID, Password et Access :

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

Chargez une table comportant 10 000 lignes. Le champ A contiendra le numéro de l'enregistrement lu (1,2,3,4,5...) tandis que le champ B contiendra un nombre aléatoire compris entre 0 et 1 :

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

La parenthèse après autogenerate est autorisée, mais elle n'est pas obligatoire.

Chargement de données à partir d'une table déjà chargée
Nous commençons par charger un fichier de table délimité, que nous nommons tab1 :

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

Chargez les champs de la table tab1 déjà chargée sous tab2 :

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

Chargez les champs de la table tab1 déjà chargée mais uniquement les enregistrements où le champ A est supérieur au champ B :

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

Chargez les champs de la table tab1 déjà chargée en les classant d'après le champ A :

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

Chargez les champs de la table tab1 déjà chargée, en les classant d'après le premier champ, puis le deuxième champ :

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

Chargez les champs de la table tab1 déjà chargée en les classant d'après le champ C par ordre décroissant, puis d'après le champ B par ordre croissant, et enfin d'après le premier champ par ordre décroissant :

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

Chargement de données à partir de champs déjà chargés
Chargez le champ Types de la table Characters déjà chargée sous A :

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

Chargement de données à partir d'une table ultérieure (instruction load antérieure)
Chargez les champs A, B et les champs calculés X et Y à partir de la table Table1 chargée dans l'instruction **SELECT** suivante :

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

Groupement de données
Chargez les champs groupés (agrégés) par numéro d'article (ArtNo) :

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

Chargez les champs groupés (agrégés) par semaine et par numéro d'article (Week et ArtNo) :

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

Lecture répétée d'un enregistrement
Dans cet exemple, soit un fichier d'entrée nommé Grades.csv contenant les notes des étudiants condensées dans un champ :

```
Student,Grades
Mike,5234
John,3345
Pete,1234
Paul,3352
```

Les notes, comprises sur une échelle de 1 à 5, représentent les matières Math, English, Science et History (maths, anglais, sciences et histoire). Nous pouvons séparer les notes en valeurs distinctes en lisant chaque enregistrement plusieurs fois à l'aide d'une clause **while** au moyen de la fonction de décompte **IterNo( )**. Lors de chaque lecture, la note est extraite au moyen de la fonction **Mid** et stockée dans Grade tandis que la matière est sélectionnée à l'aide de la fonction **pick** et conservée dans Subject. La clause **while** finale contient le test permettant de vérifier que toutes les notes ont été lues (quatre par étudiant dans ce cas), ce qui signifie que l'enregistrement d'étudiant suivant doit être lu.

```
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));
```

Le résultat est une table contenant les données suivantes :

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

Chargement de données à partir de connexions analytiques
Les échantillons de données suivants sont utilisés.

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

Chargement de données à l'aide d'une fonction

Dans les exemples suivants, nous partons du principe que le plug-in de connexions analytiques intitulé *P* contient une fonction personnalisée *Calculate(Parameter1, Parameter2)*. La fonction renvoie la table *Results* qui contient les champs *Field1* et *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```

Chargez tous les champs renvoyés lors de l'envoi des champs A et C à la fonction.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Chargez uniquement le champ Field1 lors de l'envoi des champs A et C à la fonction.

```
Load * Extension P.Calculate( Values );
```
Chargez tous les champs renvoyés lors de l'envoi des champs A et B à la fonction. Comme les champs ne sont pas spécifiés, A et B sont utilisés, car ce sont les premiers affichés dans l'ordre de la table.

```
Load * Extension P.Calculate( Values {C, C});
```
Chargez tous les champs renvoyés lors de l'envoi du champ C aux deux paramètres de la fonction.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Chargez tous les champs renvoyés lors de l'envoi à la fonction du champ A converti de force en chaîne et du champ B converti de force en valeur numérique.

Chargement de données via l'évaluation d'un script

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Chargez la table renvoyée par le script q lors de l'envoi des valeurs de A et B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Chargez la table renvoyée par le script stocké dans la variable My_R_Script lors de l'envoi des valeurs de A et B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Chargez la table renvoyée par le script stocké dans la variable My_R_Script lors de l'envoi des valeurs de B renommées D, A et C. L'utilisation du caractère * permet d'envoyer les champs non référencés restants.

L'extension de fichier des connexions DataFiles respecte la casse. Par exemple : .qvd.

Éléments de spécification du format

Chaque élément de spécification du format définit une certaine propriété du fichier de table :

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

Jeu de caractères

Un jeu de caractères est un spécificateur de fichier destiné à l'instruction **LOAD** qui définit le jeu de caractères utilisé dans le fichier.

Les spécificateurs **ansi**, **oem** et **mac** étaient utilisés dans QlikView et fonctionnent toujours. Cependant, ils ne sont pas générés lors de la création de l'instruction **LOAD** dans Qlik Sense.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
utf8	Jeu de caractères UTF-8
unicode	Jeu de caractères Unicode
ansi	Windows, page de codes 1252
oem	DOS, OS/2, AS400 et d'autres encore
mac	Page de codes 10000
codepage is	Le spécificateur **codepage** permet d'utiliser n'importe quelle page de codes Windows en tant que valeur *N*.

Limitations :

La conversion du jeu de caractères **oem** n'est pas implémentée pour MacOS. En l'absence de spécification, c'est la page de codes 1252 qui est utilisée sous Windows.

Exemple :

```
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)
```

Voir aussi :

p *Load (page 151)*

Format de table

Le format de table est un spécificateur de fichier destiné à l'instruction **LOAD** qui définit le type de fichier. En l'absence de spécification, c'est le format de fichier *.txt* qui est utilisé.

Types de format de table

Type	Description
txt	Dans un fichier texte délimité, les colonnes de la table sont séparées par un caractère délimiteur.

Type	Description
fix	Dans un fichier d'enregistrement fixe, chaque champ a une largeur d'un certain nombre de caractères. En général, de nombreux fichiers à longueur d'enregistrement fixe contiennent des enregistrements séparés par un saut de ligne. Il existe cependant des options plus avancées permettant de spécifier une taille d'enregistrement en octets ou d'étendre l'enregistrement sur plus d'une ligne grâce à **Record is**. *Si les données contiennent des caractères de plusieurs octets, l'alignement des sauts de champ peut devenir incorrect, car le format est basé sur une longueur fixe en octets.*
dif	Dans un fichier *.dif* (Data Interchange Format), un format spécial est utilisé pour définir la table.
biff	Qlik Sense peut également interpréter les données des fichiers Excel standard à l'aide du format *biff* (Binary Interchange File Format).
ooxml	Excel 2007 et les versions ultérieures utilisent le format ooxml *.xslx*.
html	Si la table fait partie d'une page ou d'un fichier html, vous devez utiliser html.
xml	xml (Extensible Markup Language) est un langage de balisage commun utilisé pour représenter des structures de données dans un format textuel.
qvd	Le format *qvd* est le format propriétaire des fichiers QVD exportés à partir d'une application Qlik Sense.
qvx	*qvx* est un format de fichier/flux utilisé pour obtenir un résultat de qualité supérieure dans Qlik Sense.

Delimiter is

Pour les fichiers de table délimités, il est possible d'indiquer un délimiteur arbitraire à l'aide du spécificateur **delimiter is**. Ce spécificateur est uniquement pertinent dans le cas de fichiers .txt délimités.

Syntaxe :

```
delimiter is char
```

Arguments :

Arguments

Argument	Description
char	Spécifie un caractère unique parmi les 127 ASCII existants.

Les valeurs suivantes peuvent également être utilisées :

Valeurs en option

Valeur	Description
'\t'	représentant un signe de tabulation, avec ou sans guillemets.
'\\'	représentant une barre oblique inverse (\).
'spaces'	représentant toutes les combinaisons d'un ou de plusieurs espaces. Les caractères non imprimables dotés d'une valeur ASCII inférieure à 32, à l'exception de CR et LF, sont interprétés comme des espaces.

En l'absence de toute spécification, **delimiter is ','** est utilisé.

Exemple :

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

Voir aussi :

p *Load (page 151)*

No eof

Le spécificateur **no eof** permet d'ignorer le caractère de fin de fichier lors du chargement de fichiers **.txt** délimités.

Syntaxe :

```
no eof
```

Si le spécificateur **no eof** est utilisé, les caractères du point de code 26, qui sinon indique une fin de fichier, sont ignorés et peuvent faire partie d'une valeur de champ.

Ce spécificateur est uniquement pertinent dans le cas de fichiers texte délimités.

Exemple :

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

Voir aussi :

p *Load (page 151)*

Labels

Labels est un spécificateur de fichier destiné à l'instruction **LOAD** qui définit l'emplacement des noms de champ dans un fichier.

Syntaxe :

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

Les noms des champs peuvent figurer en différents endroits du fichier. Si le premier enregistrement contient les noms des champs, vous devez utiliser **embedded labels**. Si aucun nom de champ n'est présent, utilisez **no labels**. Dans les fichiers *dif*, une section d'en-tête distincte, comprenant des noms de champ explicites, est quelquefois utilisée. Dans ce cas, il est nécessaire d'utiliser **explicit labels**. En l'absence de toute spécification, **embedded labels** est utilisé, y compris pour les fichiers *dif*.

Example 1:

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

Example 2:

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

Voir aussi :

p *Load (page 151)*

Header is

Spécifie la taille de l'en-tête dans les fichiers de table. Vous pouvez spécifier une longueur d'en-tête arbitraire grâce au spécificateur **header is**. Un en-tête est une section de texte qui n'est pas utilisée par Qlik Sense.

Syntaxe :

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

La longueur de l'en-tête peut être spécifiée en octets (**header is n**) ou en lignes (**header is line** ou **header is n lines**). **n** doit être un entier positif représentant la longueur de l'en-tête. En l'absence de toute spécification, **header is 0** est utilisé. Le spécificateur **header is** s'applique uniquement à un fichier de table.

Exemple :

Exemple de table de source de données contenant une ligne de texte d'en-tête à ne pas interpréter comme des données par Qlik Sense.

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

La première ligne ne sera pas chargée comme données si le spécificateur **header is 1 lines** est utilisé. Dans l'exemple, le spécificateur **embedded labels** indique à Qlik Sense d'interpréter la première ligne non exclue comme une ligne contenant des étiquettes de champ.

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

Il en résulte une table à deux champs : Col1 et Col2.

Voir aussi :

p *Load (page 151)*

Record is

Pour les fichiers à longueur d'enregistrement fixe, vous devez indiquer la longueur d'enregistrement à l'aide du spécificateur **record is**.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
n	Spécifie la longueur de l'enregistrement en octets.
line	Spécifie la longueur de l'enregistrement sous forme d'une ligne.
n lines	Spécifie la longueur de l'enregistrement en plusieurs lignes où n est un entier positif représentant la longueur de l'enregistrement.

Limitations :

Le spécificateur **record is** s'applique uniquement aux fichiers **fix**.

Voir aussi :

p *Load (page 151)*

Quotes

Quotes est un spécificateur de fichier destiné à l'instruction **LOAD** qui précise s'il est possible d'utiliser des guillemets et indique l'ordre de priorité entre les guillemets et les séparateurs. Ce spécificateur s'applique uniquement aux fichiers texte.

Syntaxe :

```
no quotes
msq
```

Si le spécificateur est omis, les guillemets standard sont employés ; autrement dit, les guillemets " " ou ' ' peuvent être utilisés, mais uniquement s'ils correspondent au premier et au dernier caractères non vides d'une valeur de champ.

Arguments :

Arguments

Argument	Description
no quotes	Utilisé si les guillemets ne sont pas acceptés dans un fichier texte.
msq	Permet de spécifier la mise entre guillemets de style moderne, autorisant le contenu multiligne dans les champs. Les champs contenant des caractères de fin de ligne doivent être placés entre guillemets doubles. L'option msq présente une limitation, à savoir que les caractères de guillemet double seuls (") figurant en tant que premier ou dernier caractère dans le contenu d'un champ sont interprétés comme début ou comme fin d'un contenu multiligne, ce qui peut aboutir à des résultats imprévus dans l'ensemble de données chargé. Dans ce cas, il est recommandé d'utiliser plutôt la mise entre guillemets standard, en omettant le spécificateur.

XML

Ce spécificateur de script s'utilise lorsque des fichiers xml sont chargés. Les options valides pour le spécificateur **XML** sont indiquées dans la syntaxe.

Il est impossible de charger des fichiers DTD dans Qlik Sense.

Syntaxe :

```
xmlsimple
```

Voir aussi :

p *Load (page 151)*

KML

Ce spécificateur de script s'emploie lors du chargement de fichiers KML à utiliser dans une visualisation de carte.

Syntaxe :

```
kml
```

Le fichier KML peut désigner des données de zones (par exemple, des pays ou des régions) représentées par des polygones, des données de lignes (comme des chemins ou des routes) ou encore des données de points (par exemple, des villes ou des lieux) représentées par des points sous la forme [long, lat].

URL is

Ce spécificateur de script permet de définir l'URL d'une connexion de données de type Fichier Web lorsque vous chargez un fichier Web.

Syntaxe :

```
URL is string
```

Arguments :

Arguments

Argument	Description
string	Indique l'URL du fichier à charger. Cet argument remplace l'URL définie dans la connexion de fichier Web utilisée.

Limitations :

Le spécificateur **URL is** s'applique uniquement aux fichiers Web. Vous devez utiliser une connexion de données de type Fichier Web existante.

Voir aussi :

p *Load (page 151)*

userAgent is

Ce spécificateur de script permet de définir l'agent utilisateur de navigateur lorsque vous chargez un fichier Web.

Syntaxe :

```
userAgent is string
```

Arguments :

Arguments

Argument	Description
string	Indique la chaîne d'agent utilisateur de navigateur. Cet argument remplacera l'agent utilisateur de navigateur par défaut "Mozilla/5.0".

Limitations :

Le spécificateur **userAgent is** s'applique uniquement aux fichiers Web.

Voir aussi :

p *Load (page 151)*

Let

L'instruction **let** complète l'instruction **set**, utilisée pour définir des variables de script. Contrairement à l'instruction **set**, l'instruction **let** évalue l'expression située à droite du signe '=' lors de l'exécution du script avant qu'elle soit attribuée à la variable.

Syntaxe :

```
Let variablename=expression
```

Exemples et résultats :

Exemple	Résultat

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` est évalué en tant que ' 3+4'.
	
	`$(y)` est évalué en tant que ' 7'.
	
	`$(z)` est évalué en tant que ' 8'.
	
	Notez la différence entre les instructions **Set** et **Let**. L'instruction **Set** assigne la chaîne '3+4' à la variable, tandis que l'instruction **Let** évalue la chaîne et assigne 7 à la variable.

Let T=now( );	`$(T)` prend la valeur de l'heure active.

Loosen Table

Vous pouvez déclarer explicitement une ou plusieurs tables de données internes Qlik Sense comme déconnectées lors de l'exécution du script grâce à l'instruction **Loosen Table**. Lorsqu'une table est déconnectée, toutes les associations entre les valeurs de champ de la table sont supprimées. Il est possible d'obtenir un effet similaire en chargeant chaque champ de la table déconnectée comme une table indépendante, sans lien. L'emploi de tables déconnectées peut s'avérer pratique lors des tests afin d'isoler temporairement différentes parties de la structure de données. Dans le visionneur de tables, une table déconnectée est signalée par des lignes en pointillé. L'utilisation d'une ou de plusieurs instructions **Loosen Table** dans le script permet à Qlik Sense d'ignorer toute déconnexion de tables définie avant l'exécution du script.

Syntaxe :

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

Les syntaxes **Loosen Table** et **Loosen Tables** sont toutes deux possibles.

*Si Qlik Sense trouve dans la structure des données des références circulaires qu'il est impossible de résoudre à l'aide de tables déclarées déconnectées de façon interactive ou explicite dans le script, une ou plusieurs tables supplémentaires seront déconnectées de force jusqu'à ce qu'il ne reste plus de références circulaires. Lorsque ce cas se produit, la boîte de dialogue **Avertissement de boucle** émet un avertissement.*

Exemple :

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

Map

L'instruction **map ... using** permet de mapper une valeur de champ ou une expression donnée aux valeurs d'une table de mappage précise. La table de mappage est créée par l'instruction **Mapping**.

Syntaxe :

```
Map fieldlist Using  mapname
```

Le mappage automatique s'applique aux champs chargés après l'instruction **Map ... Using** jusqu'à la fin du script ou jusqu'à ce qu'une instruction **Unmap** soit rencontrée.

Le mappage est effectué en dernier dans la chaîne des événements qui conduisent au stockage du champ dans la table interne de Qlik Sense. Cela signifie que le mappage n'est pas effectué à chaque fois qu'un nom de champ est rencontré dans une expression, mais plutôt lorsque la valeur est stockée sous ce nom de champ dans la table interne. Si le mappage au niveau de l'expression est requis, la fonction **Applymap ()** doit être utilisée à la place.

Arguments :

Arguments

Argument	Description
fieldlist	Liste des champs, séparés par des virgules, qui doivent être mappés à partir de cet endroit du script. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.
mapname	Nom d'une table de mappage lue précédemment dans une instruction **mapping load** ou **mapping select**.

Exemples et résultats :

Exemple	Résultat
Map Country Using Cmap;	Active le mappage du champ Country en utilisant Cmap.
Map A, B, C Using X;	Active le mappage des champs A, B et C en utilisant X.
Map * Using GenMap;	Active le mappage de tous les champs en utilisant GenMap.

NullAsNull

L'instruction **NullAsNull** permet de désactiver la conversion des valeurs NULL en valeurs de chaîne définies précédemment au moyen d'une instruction **NullAsValue**.

Syntaxe :

```
NullAsNull *fieldlist
```

L'instruction **NullAsValue** fonctionne comme un commutateur ; utilisez une instruction **NullAsValue** ou **NullAsNull** pour l'activer ou la désactiver plusieurs fois dans le script.

Arguments :

Arguments

Argument	Description
*fieldlist	Liste des champs séparés par des virgules et pour lesquels l'instruction **NullAsNull** doit être activée. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.

Exemple :

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

L'instruction **NullAsValue** spécifie les champs pour lesquels il est nécessaire de convertir en valeur la valeur NULL rencontrée.

Syntaxe :

```
NullAsValue *fieldlist
```

Par défaut, Qlik Sense considère que les valeurs NULL sont des entités manquantes ou non définies. Cependant, certains contextes de bases de données impliquent que les valeurs NULL soient considérées comme des valeurs spéciales plutôt que de simples valeurs manquantes. Il est ainsi possible de suspendre l'interdiction faite aux valeurs NULL d'être liées à d'autres valeurs NULL grâce à l'instruction **NullAsValue**.

L'instruction **NullAsValue** fonctionne comme une option et s'applique aux instructions de chargement ultérieures. Vous pouvez la désactiver à nouveau au moyen de l'instruction **NullAsNull**.

Arguments :

Arguments

Argument	Description
*fieldlist	Liste des champs séparés par des virgules et pour lesquels l'instruction **NullAsValue** doit être activée. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.

Exemple :

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

L'instruction **Qualify** permet d'activer la qualification des noms de champ, autrement dit les noms de champ se voient octroyer le nom de la table comme préfixe.

Syntaxe :

```
Qualify *fieldlist
```

La jointure automatique entre des champs portant le même nom dans des tables différentes peut être suspendue au moyen de l'instruction **qualify** qui qualifie le nom du champ par son nom de table. Les champs ainsi qualifiés sont renommés lorsque le script les détecte dans une table. Le nouveau nom suit la forme *tablename.fieldname*. *Tablename* correspond à l'étiquette de la table active ou, si aucune étiquette n'existe, au nom figurant après **from** dans les instructions **LOAD** et **SELECT**.

La qualification s'applique à tous les champs chargés après l'instruction **qualify**.

La qualification est toujours désactivée par défaut au début de l'exécution du script. La qualification d'un nom de champ peut être activée à tout moment à l'aide d'une l'instruction **qualify**. Elle peut aussi être désactivée à tout moment à l'aide d'une instruction **Unqualify**.

*L'instruction **qualify** ne doit pas être utilisée parallèlement à un rechargement partiel.*

Arguments :

Arguments

Argument	Description
*fieldlist	Liste des champs séparés par des virgules et pour lesquels la qualification doit être activée. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.

Example 1:

```
Qualify B;
LOAD A,B from x.csv;
LOAD A,B from y.csv;
```

Les deux tables **x.csv** et **y.csv** sont uniquement associées via **A**. Trois champs en résulteront : A, x.B et y.B.

Example 2:

Dans une base de données inconnue, il est généralement utile de s'assurer d'abord que seuls un ou quelques champs sont associés, comme l'illustre cet exemple :

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Seul le champ **TransID** est utilisé pour associer les tables *tab1*, *tab2* et *tab3*.

Rem

L'instruction **rem** permet d'insérer des remarques ou des commentaires dans le script ou de désactiver temporairement des instructions de script sans pour autant les supprimer.

Syntaxe :

```
Rem string
```

Tout ce qui est compris entre **rem** et le point-virgule **;** suivant est considéré comme un commentaire.

Il existe deux méthodes pour insérer des commentaires dans le script :

1. Vous pouvez créer un commentaire n'importe où dans le script (excepté entre deux guillemets) en plaçant la section en question entre **/*** et ***/**.
2. Lorsque vous tapez **//** dans le script, tout le texte situé à droite sur la même ligne devient un commentaire. (Vous remarquerez l'exception //: utilisable comme partie d'une adresse Internet.)

Arguments :

Arguments

Argument	Description
string	Texte arbitraire.

Exemple :

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

Rename

Le mot-clé de script **Rename** permet de renommer des tables ou des champs déjà chargés.

Rename field

Cette fonction de script permet de renommer un ou plusieurs champs Qlik Sense existants après leur chargement.

Il est déconseillé d'utiliser le même nom pour un champ et une fonction dans Qlik Sense

Les syntaxes **rename field** et **rename fields** sont toutes deux possibles.

Syntaxe :

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

Arguments :

Argument	Description
mapname	Nom d'une table de mappage déjà chargée qui contient une ou plusieurs paires d'anciens et de nouveaux noms de champ.
oldname	Ancien nom du champ.
newname	Nouveau nom du champ.

Limitations :

Vous ne pouvez pas renommer deux champs à l'identique.

Example 1:

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

Example 2:

```
FieldMap:
Mapping SQL SELECT oldnames, newnames from datadictionary;
Rename Fields using FieldMap;
```

Rename table

Cette fonction de script permet de renommer une ou plusieurs tables internes Qlik Sense existantes après leur chargement.

Les syntaxes **rename table** et **rename tables** sont toutes deux possibles.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
mapname	Nom d'une table de mappage déjà chargée qui contient une ou plusieurs paires d'anciens et de nouveaux noms de table.
oldname	Ancien nom de la table.
newname	Nouveau nom de la table.

Limitations :

Il est impossible de renommer de la même manière deux tables portant des noms différents. Le script génère une erreur si vous tentez de renommer une table en utilisant le nom d'une table existante.

Example 1:

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

Example 2:

```
TabMap:
Mapping LOAD oldnames, newnames from tabnames.csv;
Rename Tables using TabMap;
```

Search

L'instruction **Search** permet d'inclure ou d'exclure des champs dans la fonction de recherche intelligente.

Syntaxe :

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Vous pouvez utiliser plusieurs instructions Search pour affiner la sélection des champs à inclure dans la recherche. Les instructions sont évaluées de haut en bas.

Arguments :

Arguments

Argument	Description
*fieldlist	Liste de champs, séparés par des virgules, à inclure ou exclure des recherches effectuées avec la fonction de recherche intelligente. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.

Exemple :

Exemples de recherche

Instruction	Description
`Search Include *;`	Inclut tous les champs dans les recherches effectuées avec la fonction de recherche intelligente.
`Search Exclude [*ID];`	Exclut tous les champs se terminant par ID des recherches effectuées avec la fonction de recherche intelligente.
`Search Exclude '*ID';`	Exclut tous les champs se terminant par ID des recherches effectuées avec la fonction de recherche intelligente.
`Search Include ProductID;`	Inclut le champ ProductID dans les recherches effectuées avec la fonction de recherche intelligente.

La combinaison de ces trois instructions, dans cet ordre, entraîne l'exclusion de tous les champs se terminant par ID des recherches effectuées avec la fonction de recherche intelligente, à l'exception de ProductID.

Section

L'instruction **section** permet de déterminer si les instructions **LOAD** et **SELECT** ultérieures doivent être considérées comme des données ou comme une définition des droits d'accès.

Syntaxe :

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

En l'absence de toute spécification, **section application** est utilisé. La définition de **section** est valable jusqu'à ce qu'une nouvelle instruction **section** soit créée.

Exemple :

```
Section access;
Section application;
```

Select

La sélection de champs à partir d'une source de données ODBC ou d'un fournisseur OLE DB s'effectue au moyen d'instructions **SELECT** SQL standard. Cependant, l'acceptation des instructions **SELECT** dépend du pilote ODBC ou du fournisseur OLE DB utilisé. L'utilisation de l'instruction **SELECT** requiert une connexion de données ouverte vers la source.

Syntaxe :

```
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 ]
```

Il est par ailleurs possible, dans certains cas, de concaténer plusieurs instructions **SELECT** en une seule à l'aide d'un opérateur **union** :

```
selectstatement Union selectstatement
```

L'instruction **SELECT** est interprétée par le pilote ODBC ou le fournisseur OLE DB. Il peut donc arriver que des écarts par rapport à la syntaxe SQL générale se produisent, suivant les capacités des pilotes ODBC ou du fournisseur OLE DB, par exemple :

- **as** n'est pas toujours autorisé, autrement dit *aliasname* doit suivre immédiatement *fieldname*.
- **as** est parfois obligatoire si un nom d'alias (*aliasname*) est utilisé.
- **distinct**, **as**, **where**, **group by**, **order by** et **union** ne sont pas toujours pris en charge.
- Le pilote ODBC n'accepte pas toujours tous les types de guillemets indiqués ci-dessus.

*Ceci n'est pas une description complète de l'instruction SQL **SELECT**. Les instructions **SELECT** peuvent, par exemple, être imbriquées. Il est aussi possible d'effectuer plusieurs jointures dans une instruction **SELECT**, le nombre de fonctions autorisées dans les expressions est parfois très grand, etc.*

Arguments :

Arguments

Argument	Description
distinct	**distinct** est un prédicat utilisé si les combinaisons de valeurs en double dans les champs sélectionnés ne doivent être chargées qu'une seule fois.
distinctrow	**distinctrow** est un prédicat utilisé si les enregistrements en double dans la table source ne doivent être chargés qu'une seule fois.
fieldlist	**`fieldlist ::= (*| field ) {, field }`** Liste des champs à sélectionner. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs de la table. **`fieldlist ::= field {, field }`** Liste d'un ou de plusieurs champs, séparés par des virgules. **`field ::= ( fieldref| expression ) [as aliasname ]`** L'expression peut, par exemple, être une fonction numérique ou une fonction de chaîne basée sur un ou plusieurs autres champs. Certains des opérateurs et des fonctions généralement acceptés sont : +, -, *, /, & (concaténation de chaînes), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname), etc. Pour plus d'informations, consultez la documentation du pilote ODBC. **`fieldref ::= [ tablename. ] fieldname`** **tablename** et **fieldname** sont des chaînes de texte identiques à ce qu'elles représentent. Elles doivent être mises entre guillemets doubles droits si elles contiennent des espaces, par exemple. La clause **as** est utilisée pour donner un nouveau nom au champ.
from	**`tablelist ::= table {, table }`** Liste des tables à partir desquelles les champs doivent être sélectionnés. **`table ::= tablename [ [as ] aliasname ]`** La chaîne **tablename** ne doit pas obligatoirement être mise entre guillemets.
where	**where** est une clause utilisée pour indiquer si un enregistrement doit être inclus ou pas dans la sélection. **criterion** est une expression logique qui peut parfois être très complexe. Certains des opérateurs acceptés sont : opérateurs et fonctions mathématiques, **=**, **<>** ou **#(**non égal), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** ainsi que les nouvelles instructions **SELECT**. Pour plus d'informations, consultez la documentation du pilote ODBC ou du fournisseur OLE DB.

Argument	Description
group by	**group by** est une clause utilisée pour agréger (grouper) plusieurs enregistrements en un seul. Dans un groupe, pour un champ donné, soit tous les enregistrements ont la même valeur, soit le champ ne peut être utilisé que dans une expression, comme par exemple une somme ou une moyenne. L'expression basée sur un ou plusieurs champs est définie dans l'expression du symbole de champ.
having	**having** est une clause utilisée pour qualifier des groupes comme la clause **where** sert à qualifier des enregistrements.
order by	**order by** est une clause utilisée pour spécifier l'ordre de tri de la table obtenue par l'instruction **SELECT**.
join	**join** est un qualificateur indiquant que plusieurs tables doivent être jointes en une seule. Les noms des champs et des tables doivent être mis entre guillemets s'ils contiennent des espaces vides ou des lettres de jeux de caractères nationaux. Lorsque le script est généré automatiquement par Qlik Sense, les guillemets utilisés sont les guillemets favoris du pilote ODBC ou du fournisseur OLE DB spécifié dans la définition de la source de données de l'instruction **Connect**.

Example 1:

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

Example 2:

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

Example 3:

```
SELECT `Order ID`, `Product ID`,

`Unit Price` * Quantity * (1-Discount) as NetSales

FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,

Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`

FROM `Order Details`, Orders

where Orders.`Order ID` = `Order Details`.`Order ID`

group by `Order Details`.`Order ID`;
```

Set

L'instruction **set** permet de définir des variables de script. Ces variables peuvent servir à remplacer des chaînes, des chemins d'accès, des lecteurs, etc.

Syntaxe :

```
Set variablename=string
```

Example 1:

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

Example 2:

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

Example 3:

```
Set BudgetYear=2012;
```

Sleep

L'instruction **sleep** interrompt l'exécution du script pendant la durée spécifiée.

Syntaxe :

```
Sleep n
```

Arguments :

Argument	Description
n	Spécifié en millisecondes, où *n* est un entier positif inférieur ou égal à 3600000 (c.-à-d. 1 heure). La valeur peut être une expression.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

L'instruction **SQL** vous permet d'envoyer une commande SQL arbitraire via une connexion ODBC ou OLE DB.

Syntaxe :

```
SQL sql_command
```

L'envoi d'instructions SQL qui mettent à jour la base de données entraîne le renvoi d'une erreur si Qlik Sense a ouvert la connexion ODBC en mode lecture seule.

La syntaxe :

```
SQL SELECT * from tab1;
```
est autorisée et est , par souci de cohérence, la syntaxe privilégiée pour **SELECT**. Le préfixe SQL reste cependant facultatif pour les instructions **SELECT**.

Arguments :

Argument	Description
sql_command	Commande SQL valide.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

L'instruction **sqlcolumns** renvoie un ensemble de champs qui décrit les colonnes d'une source de données ODBCou OLE DB à laquelle une instruction **connect** a été adressée.

Syntaxe :

```
SQLcolumns
```

Il est possible de combiner les champs aux champs générés par les commandes **sqltables** et **sqltypes** afin d'obtenir une vue d'ensemble satisfaisante d'une base de données particulière. Les douze champs standard sont les suivants :

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Pour une description détaillée de ces champs, consultez un manuel de référence sur ODBC.

Exemple :

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Il est possible que certains pilotes ODBC ne prennent pas en charge cette commande. Certains pilotes ODBC peuvent générer des champs supplémentaires.

SQLTables

L'instruction **sqltables** renvoie un ensemble de champs qui décrit les tables d'une source de données ODBC ou OLE DB à laquelle une instruction **connect** a été adressée.

Syntaxe :

```
SQLTables
```

Il est possible de combiner les champs aux champs générés par les commandes **sqlcolumns** et **sqltypes** afin d'obtenir une vue d'ensemble satisfaisante d'une base de données particulière. Les cinq champs standard sont les suivants :

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Pour une description détaillée de ces champs, consultez un manuel de référence sur ODBC.

Exemple :

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Il est possible que certains pilotes ODBC ne prennent pas en charge cette commande. Certains pilotes ODBC peuvent générer des champs supplémentaires.

SQLTypes

L'instruction **sqltypes** renvoie un ensemble de champs décrivant les types d'une source de données ODBC ou OLE DB à laquelle une instruction **connect** a été adressée.

Syntaxe :

```
SQLTypes
```

Il est possible de combiner les champs aux champs générés par les commandes **sqlcolumns** et **sqltables** afin d'obtenir une vue d'ensemble satisfaisante d'une base de données particulière. Les quinze champs standard sont les suivants :

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

Pour une description détaillée de ces champs, consultez un manuel de référence sur ODBC.

Exemple :

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Il est possible que certains pilotes ODBC ne prennent pas en charge cette commande. Certains pilotes ODBC peuvent générer des champs supplémentaires.

Star

Vous pouvez utiliser l'instruction **star** pour définir la chaîne devant représenter l'ensemble des valeurs d'un champ dans la base de données. Elle affecte les instructions **LOAD** et **SELECT** ultérieures.

Syntaxe :

```
Star is[ string ]
```

Arguments :

Arguments

Argument	Description
string	Texte arbitraire. Notez que la chaîne doit être mise entre guillemets si elle contient des espaces. À défaut de toute indication, **star is;** est utilisé. En d'autres termes, aucun astérisque n'est disponible à moins d'être spécifié de manière explicite. Cette définition est valable jusqu'à ce qu'une nouvelle instruction **star** soit créée.

L'utilisation de l'instruction **Star is** n'est pas recommandée dans les données qui font partie du script (sous **Section Application**) si l'accès de section est utilisé. L'astérisque est néanmoins totalement pris en charge par les champs protégés dans la partie **Section Access** du script. Dans ce cas, il est inutile d'utiliser l'instruction explicite **Star is**, puisqu'elle est toujours implicite dans l'accès de section.

Limitations

- Vous ne pouvez pas utiliser l'astérisque avec les champs clés, c'est-à-dire les champs qui lient des tables.
- L'astérisque n'est pas non plus compatible avec les champs affectés par l'instruction **Unqualify**, car cela pourrait influer sur des champs qui lient des tables.
- Vous ne pouvez pas utiliser l'astérisque avec des tables non logiques, par exemple des tables info-load ou mapping-load.
- Lorsque l'astérisque est utilisé dans un champ de réduction (un champ qui établit la liaison avec les données) dans l'accès de section, il représente les valeurs répertoriées dans ce champ au sein de l'accès de section. Il ne représente pas d'autre valeurs éventuellement présentes dans les données, mais qui ne figurent pas dans l'accès de section.
- Il n'est pas possible d'utiliser l'astérisque avec des champs affectés par une forme de réduction de données, quelle qu'elle soit, en dehors de la zone **Section Access**.

Exemple

L'exemple ci-dessous est un extrait d'un script de chargement de données comprenant un accès de section.

```
Star is *;
```

```
Section Access;

LOAD * INLINE [

ACCESS, USERID, OMIT

ADMIN, ADMIN,

USER, USER1, SALES

USER, USER2, WAREHOUSE

USER, USER3, EMPLOYEES

USER, USER4, SALES

USER, USER4, WAREHOUSE

USER, USER5, *

];


Section Application;

LOAD * INLINE [

SALES, WAREHOUSE, EMPLOYEES, ORDERS

1, 2, 3, 4

];
```

Les conditions suivantes s'appliquent :

- Le signe *Star* correspond à *.
- L'utilisateur *ADMIN* voit tous les champs. Rien n'est omis.
- L'utilisateur *USER1* n'est pas en mesure de voir le champ *SALES*.
- L'utilisateur *USER2* n'est pas en mesure de voir le champ *WAREHOUSE* .
- L'utilisateur *USER3* ne peut pas voir le champ *EMPLOYEES*.
- L'utilisateur *USER4* est ajouté deux fois à la solution afin d'omettre deux champs pour cet utilisateur via OMIT, *SALES* et *WAREHOUSE*.
- *USER5* présente un "*" en plus, ce qui signifie que tous les champs répertoriés dans OMIT sont indisponibles, à savoir, l'utilisateur *USER5* ne peut pas voir les champs *SALES*, *WAREHOUSE* ni *EMPLOYEES* , mais cet utilisateur peut voir le champ *ORDERS*.

Store

L'instruction **Store** crée un fichier QVD, CSV ou text.

Syntaxe :

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

L'instruction crée un fichier QVD, CSV ou TXT nommé de manière explicite.

L'instruction permet uniquement d'exporter des champs provenant d'une table de données. Si vous devez exporter des champs issus de plusieurs tables, définissez au préalable une jointure (join) explicite dans le script afin de créer la table de données à exporter.

Les valeurs de texte sont exportées vers le fichier CSV au format UTF-8. Vous pouvez spécifier un délimiteur (voir **LOAD**). L'instruction **store** envoyée à un fichier CSV ne prend pas en charge l'exportation BIFF.

Arguments :

Arguments de la commande Store

Argument	Description
*fieldlist::= (* | field) { , field })*	Liste des champs à sélectionner. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. *field::= fieldname [***as** *aliasname]* *fieldname* est un texte identique à un nom de champ dans la *table*. (Notez que le nom du champ doit être mis entre guillemets doubles droits ou entre crochets s'il contient des espaces ou d'autres caractères non standard.) *aliasname* est un nom alternatif à utiliser pour le champ dans le fichier QVD ou CSV résultant.
table	Étiquette de script représentant une table déjà chargée, à utiliser comme source pour les données.

Argument	Description
filename	Nom du fichier cible incluant un chemin d'accès valide à une connexion de données de type dossier existante. **Exemple :** ***'lib://Table Files/target.qvd'*** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data\sales.qvd* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data\sales.qvd* Si le chemin d'accès est omis, Qlik Sense stocke le fichier dans le répertoire que lui indique l'instruction **Directory**. En l'absence d'instruction **Directory**, Qlik Sense conserve le fichier dans le répertoire de travail, *C:\Users\{user}\Documents\Qlik\Sense\Apps*.
format-spec ::=((**txt** | **qvd**))	La spécification du format comprend le texte **txt** pour les fichiers texte ou le texte **qvd** pour les fichiers qvd. Si le format n'est pas spécifié, **qvd** est utilisé.

Exemples :

```
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';
```

L'extension de fichier des connexions DataFiles respecte la casse. Par exemple : .qvd.

Table/Tables

Les mots-clés de script **Table** et **Tables** s'utilisent dans les instructions **Drop**, **Comment** et **Rename**, de même que comme spécificateur de format dans les instructions **Load**.

Tag

Cette instruction de script permet d'assigner des balises à un ou plusieurs champs ou tables. Si vous tentez d'ajouter une balise à un champ ou une table qui ne figure pas dans l'application, l'opération est ignorée. Si des occurrences de nom de champ ou de balise conflictuelles sont détectées, la dernière valeur est utilisée.

Syntaxe :

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Arguments

Argument	Description
fieldlist	Un ou plusieurs champs à baliser dans une liste séparée par des virgules.
mapname	Nom d'une table de mappage chargée précédemment dans une instruction **mapping Load** ou **mapping Select**.
tablelist	Liste séparée par des virgules des tables à baliser.
tagname	Nom de la balise à appliquer au champ.

Example 1:

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

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

L'instruction **trace** écrit une chaîne dans la fenêtre **Progression de l'exécution du script** et, le cas échéant, la consigne dans le fichier journal du script. Elle s'avère extrêmement pratique pour le débogage. L'utilisation d'expansions $ de variables calculées avant l'instruction **trace** vous permet de personnaliser le message.

Syntaxe :

```
Trace string
```

Example 1:

L'instruction suivante peut être utilisée juste après l'instruction Load qui charge la table 'Main' (principale).

```
Trace Main table loaded;
```

Cela affiche le texte 'Main table loaded' (Table principale chargée) dans la boîte de dialogue d'exécution du script et dans le fichier journal.

Example 2:

Les instructions suivantes peuvent être utilisées juste après l'instruction Load qui charge la table 'Main' (principale).

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Cela affichera un texte indiquant le nombre de lignes dans la boîte de dialogue d'exécution du script et dans le fichier journal, par exemple, '265,391 rows in Main table' (265 391 lignes dans la table principale).

Unmap

L'instruction **Unmap** désactive le mappage de valeurs de champ spécifié par une instruction **Map … Using** précédente pour les champs chargés ultérieurement.

Syntaxe :

```
Unmap *fieldlist
```

Arguments :

Arguments

Argument	Description
*fieldlist	Liste des champs séparés par des virgules et qui ne doivent plus être mappés à partir de cet endroit du script. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés.

Exemples et résultats :

Exemple	Résultat
Unmap Country;	Désactive le mappage du champ Country.
Unmap A, B, C;	Désactive le mappage des champs A, B et C.
Unmap * ;	Désactive le mappage de tous les champs.

Unqualify

L'instruction **Unqualify** permet de désactiver la qualification des noms de champ qui a été précédemment activée par l'instruction **Qualify**.

Syntaxe :

```
Unqualify *fieldlist
```

Arguments :

Arguments

Argument	Description
*fieldlist	Liste des champs séparés par des virgules et pour lesquels la qualification doit être activée. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs. Les caractères génériques * et ? sont autorisés dans les noms des champs. Il peut s'avérer nécessaire de mettre les noms des champs entre guillemets lorsque des caractères génériques sont utilisés. Pour plus d'informations, consultez la documentation relative à l'instruction **Qualify**.

Example 1:

Dans une base de données inconnue, il est généralement utile de s'assurer d'abord que seuls un ou quelques champs sont associés, comme l'illustre cet exemple :

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Tout d'abord, la qualification est activée pour tous les champs.
Ensuite, la qualification est désactivée pour **TransID**.
Seul le champ **TransID** est utilisé pour associer les tables *tab1*, *tab2* et *tab3*. Tous les autres champs seront qualifiés par le nom de table.

Untag

Cette instruction de script permet de supprimer des balises des champs ou des tables. Si vous tentez de supprimer une balise d'un champ ou d'une table qui ne figure pas dans l'application, l'opération est ignorée.

Syntaxe :

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Arguments :

Arguments

Argument	Description
fieldlist	Un ou plusieurs champs dont les balises doivent être supprimées dans une liste séparée par des virgules.
mapname	Nom d'une table de mappage déjà chargée dans une instruction mapping **LOAD** ou mapping **SELECT**.
tablelist	Liste séparée par des virgules des tables dont les balises doivent être supprimées.
tagname	Nom de la balise à supprimer du champ.

Example 1:

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

Example 2:

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

2.6 Répertoire de travail

Si vous faites référence à un fichier dans une instruction de script et que le chemin est omis, Qlik Sense recherche le fichier en suivant l'ordre ci-dessous :

1. Le répertoire spécifié par une instruction **Directory** (uniquement pris en charge en mode de script hérité).
2. En l'absence d'instruction **Directory**, Qlik Sense recherche le fichier dans le répertoire de travail.

Répertoire de travail de Qlik Sense Desktop

Dans Qlik Sense Desktop, le répertoire de travail correspond à *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Répertoire de travail de Qlik Sense

Dans une installation de serveur Qlik Sense, le répertoire de travail est spécifié dans Qlik Sense Repository Service ; par défaut, il s'agit de *C:\ProgramData\Qlik\Sense\Apps*. Consultez l'aide de Console de gestion Qlik pour plus d'informations.

2 Utilisation des variables dans l'éditeur de chargement de données

Dans Qlik Sense, une variable désigne un conteneur qui stocke un calcul ou une valeur statique, par exemple une valeur numérique ou alphanumérique. Lorsque vous utilisez la variable dans l'application, les modifications que vous lui apportez sont répercutées dans toutes ses occurrences. Vous pouvez définir des variables à partir de la vue d'ensemble des variables ou dans le script à l'aide de l'éditeur de chargement de données. Pour spécifier la valeur d'une variable, utilisez des instructions **Let** ou **Set** dans le script de chargement de données.

Vous pouvez également manipuler les variables Qlik Sense à partir de la vue d'ensemble des variables, lors de l'édition d'une feuille.

2.7 Vue d'ensemble

Si le premier caractère d'une valeur de variable est un signe égal (=), Qlik Sense tente d'évaluer la valeur comme une formule (expression Qlik Sense), puis affiche ou renvoie le résultat plutôt que le texte de la formule proprement dit.

Lorsqu'une variable est utilisée, elle est remplacée par sa valeur. Il est possible d'utiliser des variables dans le script pour les expansions $ et dans plusieurs instructions de contrôle. Cette technique s'avère très pratique lorsque la même chaîne (un chemin d'accès, par exemple) est répétée à de nombreuses reprises dans le script.

Certaines variables système particulières sont définies par Qlik Sense au début de l'exécution du script, quelle que soit leur valeur précédente.

2.8 Définition d'une variable

Les variables permettent de stocker des valeurs statiques ou le résultat d'un calcul. Lors de la définition d'une variable, utilisez la syntaxe suivante :

```
set variablename = string
```

ou

```
let variable = expression
```

L'instruction **Set** est utilisée pour l'affectation de chaîne. Elle affecte le texte à droit du signe égal à la variable. L'instruction **Let** évalue une expression à droite du signe égal lors de l'exécution du script et affecte le résultat de l'expression à la variable.

Les variables sont sensibles à la casse.

Il est déconseillé d'utiliser le même nom pour un champ et une fonction dans Qlik Sense

Exemples :

La variable `set x = 3 + 4; //` recevra comme valeur la chaîne '3 + 4'.

`let x = 3 + 4; //` renvoie la valeur 7.

`set x = Today(); //` renvoie la valeur 'Today()'.

`let x = Today(); //` renvoie la valeur de la date d'aujourd'hui, par exemple, '9/27/2021'.

2.9 Suppression d'une variable

Si vous supprimez une variable du script et rechargez les données, la variable est conservée dans l'application. Si vous souhaitez retirer complètement la variable de l'application, vous devez également la supprimer de la boîte de dialogue Variables.

2.10 Chargement d'une valeur de variable comme valeur de champ

Si vous souhaitez charger une valeur de variable comme valeur de champ dans une instruction **LOAD** et que le résultat de l'expansion de dollar est du texte plutôt qu'une valeur numérique ou une expression, vous devez placer la variable étendue entre guillemets simples.

Exemple :

Cet exemple illustre le chargement de la variable système contenant la liste d'erreurs de script dans une table. Vous pouvez noter que l'expansion de ScriptErrorCount dans la clause **If** ne nécessite pas de guillemets, contrairement à celle de ScriptErrorList.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1; END IF
```

2.11 Calcul des variables

Qlik Sense offre différentes façons d'utiliser des variables dont les valeurs sont calculées. Le résultat dépend de la façon dont elles sont définies et appelées dans une expression.

Dans cet exemple, nous chargeons des données intégrées :

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
```

```
    C, 410
    C, 330
];
```

Définissons deux variables :

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

Dans la seconde variable, un signe égal est ajouté avant l'expression. Ceci entraîne le calcul de la variable avant qu'elle ne soit étendue et l'expression évaluée.

Si vous utilisez la variable vSales telle quelle, par exemple dans une mesure, le résultat correspond à la chaîne Sum(Sales), c'est-à-dire qu'aucun calcul n'est réalisé.

Si vous ajoutez une expansion $ et appelez $(vSales) dans l'expression, la variable est étendue et la somme des ventes (Sales) est affichée.

Enfin, si vous appelez $(vSales2), la variable est calculée avant d'être étendue. En d'autres termes, le résultat affiché correspond à la somme totale des ventes (Sales). Le graphique suivant illustrant les résultats montre la différence entre l'utilisation de =$(vSales) et de =$(vSales2) comme expressions de mesure :

Résultats

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Comme vous pouvez le constater, $(vSales) aboutit à la somme partielle d'une valeur de dimension tandis que $(vSales2) donne la somme totale.

Les variables de script suivantes sont disponibles :

- *Variables d'erreur (page 269)*
- *Variables d'interprétation des nombres (page 204)*
- *Variables système (page 195)*
- *Variables de manipulation des valeurs (page 202)*

2.12 Variables système

Les variables système, dont certaines sont définies au sein du système, fournissent des informations sur le système et l'application Qlik Sense.

Vue d'ensemble des variables système

Certaines fonctions sont décrites plus en détail après la vue d'ensemble. Dans ce cas, il vous suffit de cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Floppy

Renvoie la lettre du premier lecteur de disquette trouvé, normalement *a:*. Il s'agit d'une variable définie par le système.

```
Floppy
```

Cette variable n'est pas prise en charge en mode standard.

CD

Renvoie la lettre du premier lecteur de CD-ROM trouvé. Si aucun lecteur de CD-ROM n'est trouvé, la valeur *c:* est renvoyée. Il s'agit d'une variable définie par le système.

```
CD
```

Cette variable n'est pas prise en charge en mode standard.

Include

La variable **Include/Must_Include** spécifie un fichier qui contient le texte à inclure dans le script et à évaluer comme code de script. Elle n'est pas utilisée pour ajouter des données. Il est possible de stocker des parties du code de script dans un fichier texte distinct afin de les réutiliser dans d'autres applications. Il s'agit d'une variable définie par l'utilisateur.

```
$(Include=filename)
$(Must_Include=filename)
```

HidePrefix

Tous les noms de champ commençant par cette chaîne de texte sont masqués de la même manière que les champs système. Il s'agit d'une variable définie par l'utilisateur.

```
HidePrefix
```

HideSuffix

Tous les noms de champ se terminant par cette chaîne de texte sont masqués de la même manière que les champs système. Il s'agit d'une variable définie par l'utilisateur.

```
HideSuffix
```

QvPath

Renvoie la chaîne de navigation jusqu'à l'exécutable Qlik Sense. Il s'agit d'une variable définie par le système.

```
QvPath
```

Cette variable n'est pas prise en charge en mode standard.

QvRoot

Renvoie le répertoire racine de l'exécutable Qlik Sense. Il s'agit d'une variable définie par le système.

```
QvRoot
```

Cette variable n'est pas prise en charge en mode standard.

QvWorkPath

Renvoie la chaîne de navigation jusqu'à l'application Qlik Sense active. Il s'agit d'une variable définie par le système.

```
QvWorkPath
```

Cette variable n'est pas prise en charge en mode standard.

QvWorkRoot

Renvoie le répertoire racine de l'application Qlik Sense active. Il s'agit d'une variable définie par le système.

```
QvWorkRoot
```

Cette variable n'est pas prise en charge en mode standard.

StripComments

Si cette variable est définie sur 0, le vidage sélectif des commentaires /*..*/ et // dans le script est bloqué. Si cette variable n'est pas définie, le vidage sélectif des commentaires sera toujours effectué.

```
StripComments
```

Verbatim

En général, les espaces de début et de fin (ASCII 32) de toutes les valeurs de champs sont automatiquement supprimés avant que les valeurs ne soient chargées dans la base de données Qlik Sense. La définition de cette variable sur 1 suspend le vidage sélectif des espaces. Les caractères de tabulation (ASCII 9) et les espaces insécables (ANSI 160) ne sont jamais supprimés.

```
Verbatim
```

OpenUrlTimeout

Cette variable définit le délai d'attente, exprimé en secondes, que Qlik Sense doit respecter pour récupérer les données de sources URL (par ex. HTML des pages). Si elle est omise, le délai d'attente est d'environ 20 minutes.

```
OpenUrlTimeout
```

WinPath

Renvoie la chaîne de navigation jusqu'à Windows. Il s'agit d'une variable définie par le système.

```
WinPath
```

Cette variable n'est pas prise en charge en mode standard.

WinRoot

Renvoie le répertoire racine de Windows. Il s'agit d'une variable définie par le système.

```
WinRoot
```

Cette variable n'est pas prise en charge en mode standard.

CollationLocale

Indique les paramètres régionaux à utiliser pour l'ordre de tri et les correspondances de recherche. La valeur désigne le nom de culture de paramètres régionaux spécifiques, par exemple fr-FR.Il s'agit d'une variable définie par le système.

```
CollationLocale
```

CreateSearchIndexOnReload

Cette variable définit si les fichiers d'index de recherche doivent être créés au cours du rechargement des données.

```
CreateSearchIndexOnReload
```

CreateSearchIndexOnReload

Cette variable définit si les fichiers d'index de recherche doivent être créés au cours du rechargement des données.

Syntaxe :

```
CreateSearchIndexOnReload
```

Vous pouvez définir s'il convient de créer les fichiers d'index de recherche lors du rechargement des données ou après la première requête de recherche de l'utilisateur. L'avantage de créer les fichiers d'index de recherche lors du rechargement des données est d'éviter au premier utilisateur effectuant une recherche d'attendre un certain temps avant d'obtenir les résultats. Ce laps de temps doit être évalué par rapport à la durée, plus longue, du rechargement des données que nécessite la création de l'index de recherche.

Si cette variable est omise, les fichiers d'index de recherche ne seront pas créés au cours du rechargement des données.

Pour les applications de session, les fichiers d'index de recherche ne seront pas créés lors du rechargement des données, quel que soit le paramètre de cette variable.

Example 1: Création de fichiers d'index de recherche lors du rechargement des données

```
set CreateSearchIndexOnReload=1;
```

Example 2: Création de fichiers d'index de recherche après la première requête de recherche

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Tous les noms de champ commençant par cette chaîne de texte sont masqués de la même manière que les champs système. Il s'agit d'une variable définie par l'utilisateur.

Syntaxe :

```
HidePrefix
```

Exemple :

```
set HidePrefix='_' ;
```

Si cette instruction est utilisée, les noms des champs commençant par un tiret de soulignement ne s'affichent pas dans les listes de noms de champ lorsque les champs système sont masqués.

HideSuffix

Tous les noms de champ se terminant par cette chaîne de texte sont masqués de la même manière que les champs système. Il s'agit d'une variable définie par l'utilisateur.

Syntaxe :

```
HideSuffix
```

Exemple :

```
set HideSuffix='%';
```

Si cette instruction est utilisée, les noms des champs finissant par un symbole de pourcentage ne s'affichent pas dans les listes de noms de champ lorsque les champs système sont masqués.

Include

La variable **Include/Must_Include** spécifie un fichier qui contient le texte à inclure dans le script et à évaluer comme code de script. Elle n'est pas utilisée pour ajouter des données. Il est possible de stocker des parties du code de script dans un fichier texte distinct afin de les réutiliser dans d'autres applications. Il s'agit d'une variable définie par l'utilisateur.

Cette variable prend uniquement en charge les connexions de données de type dossier en mode standard.

Syntaxe :

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Il existe deux versions de la variable :

- **Include** ne génère pas d'erreur si le fichier est introuvable ; l'échec se produit de manière silencieuse.
- **Must_Include** génère une erreur si le fichier est introuvable.

Si vous ne spécifiez aucun chemin d'accès, le nom du fichier sera relatif au répertoire de travail de l'application Qlik Sense. Vous pouvez également spécifier un chemin d'accès absolu ou un chemin d'accès à une connexion de dossier lib://. N'insérez pas d'espace avant ou après le signe égal.

*La construction **set Include =**filename n'est pas admise.*

Exemples :

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Limitations

Compatibilité croisée limitée entre les fichiers chiffrés au format UTF-8 sous Windows par rapport à Linux.

L'utilisation d'UTF-8 avec BOM (Byte Order Mark - marque d'ordre d'octet) est facultative. BOM peut interférer avec l'utilisation d'UTF-8 dans un logiciel qui n'attend pas d'octets non-ASCII au début d'un fichier, mais qui peut sinon traiter le flux de texte.

- Les systèmes Windows utilisent BOM dans UTF-8 pour identifier qu'un fichier est chiffré au format UTF-8, malgré le fait qu'il n'existe aucune ambiguïté dans le stockage des octets.
- Unix/Linux utilisent UTF-8 pour Unicode, mais ils n'utilisent pas BOM, car ce dernier interfère avec la syntaxe des fichiers de commande.

Cela a des conséquences pour Qlik Sense.

- Dans Windows, tout fichier qui commence par une marque BOM UTF-8 est considéré comme un fichier de script UTF-8. Sinon, le chiffrement ANSI est présumé.
- Dans Linux, la page de code 8 bits par défaut du système est UTF-8. C'est pourquoi le chiffrement UTF-8 fonctionne, même s'il ne contient pas de marque BOM.

Voilà pourquoi la portabilité ne peut pas être garantie. Il n'est pas toujours possible de créer un fichier sous Windows qui puisse être interprété par Linux et inversement. Il n'existe pas de compatibilité croisée entre les deux systèmes en ce qui concerne les fichiers chiffrés au format UTF-8, à cause du traitement différent de la marque BOM.

OpenUrlTimeout

Cette variable définit le délai d'attente, exprimé en secondes, que Qlik Sense doit respecter pour récupérer les données de sources URL (par ex. HTML des pages). Si elle est omise, le délai d'attente est d'environ 20 minutes.

Syntaxe :

```
OpenUrlTimeout
```

Exemple :

```
set OpenUrlTimeout=10;
```

StripComments

Si cette variable est définie sur 0, le vidage sélectif des commentaires /*..*/ et // dans le script est bloqué. Si cette variable n'est pas définie, le vidage sélectif des commentaires sera toujours effectué.

Syntaxe :

```
StripComments
```

Certains pilotes de base de données utilisent /*..*/ comme des astuces d'optimisation dans les instructions **SELECT**. Si tel est le cas, les commentaires ne doivent pas être supprimés avant l'envoi de l'instruction **SELECT** au pilote de base de données.

Il est recommandé de réinitialiser cette variable sur 1 immédiatement après la ou les instructions qui en ont besoin.

Exemple :

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

En général, les espaces de début et de fin (ASCII 32) de toutes les valeurs de champs sont automatiquement supprimés avant que les valeurs ne soient chargées dans la base de

données Qlik Sense. La définition de cette variable sur 1 suspend le vidage sélectif des espaces. Les caractères de tabulation (ASCII 9) et les espaces insécables (ANSI 160) ne sont jamais supprimés.

Syntaxe :

```
Verbatim
```

Exemple :

```
set Verbatim = 1;
```

2.13 Variables de manipulation des valeurs

Cette section décrit les variables utilisées pour gérer les valeurs NULL et d'autres valeurs.

Vue d'ensemble des variables de manipulation des valeurs

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

NullDisplay
Le symbole défini remplace toutes les valeurs NULL d'ODBC et des connecteurs au niveau le plus bas des données. Il s'agit d'une variable définie par l'utilisateur.

```
NullDisplay
```

NullInterpret
Le symbole défini est interprété comme NULL lorsqu'il est rencontré dans un fichier texte, un fichier Excel ou une instruction inline. Il s'agit d'une variable définie par l'utilisateur.

```
NullInterpret
```

NullValue
Si l'instruction **NullAsValue** est utilisée, le symbole défini remplace toutes les valeurs NULL dans les champs **NullAsValue** indiqués par la chaîne spécifiée.

```
NullValue
```

OtherSymbol
Définit un symbole à traiter comme « toutes les autres valeurs » avant une instruction **LOAD/SELECT**. Il s'agit d'une variable définie par l'utilisateur.

```
OtherSymbol
```

NullDisplay

Le symbole défini remplace toutes les valeurs NULL d'ODBC et des connecteurs au niveau le plus bas des données. Il s'agit d'une variable définie par l'utilisateur.

Syntaxe :

```
NullDisplay
```

Exemple :

```
set NullDisplay='<NULL>';
```

NullInterpret

Le symbole défini est interprété comme NULL lorsqu'il est rencontré dans un fichier texte, un fichier Excel ou une instruction inline. Il s'agit d'une variable définie par l'utilisateur.

Syntaxe :

```
NullInterpret
```

Exemples :

```
set NullInterpret=' ';
set NullInterpret =;
```

ne renvoie pas de valeur NULL pour les valeurs vides dans Excel, mais en renvoie pour un fichier texte CSV.

```
set NullInterpret ='';
```

renvoie des valeurs NULL pour les valeurs vides dans Excel.

NullValue

Si l'instruction **NullAsValue** est utilisée, le symbole défini remplace toutes les valeurs NULL dans les champs **NullAsValue** indiqués par la chaîne spécifiée.

Syntaxe :

```
NullValue
```

Exemple :

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Définit un symbole à traiter comme « toutes les autres valeurs » avant une instruction **LOAD/SELECT**. Il s'agit d'une variable définie par l'utilisateur.

Syntaxe :

```
OtherSymbol
```

Exemple :

```
set OtherSymbol='+';
LOAD * inline
```

```
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

La valeur de champ Y='b' est désormais liée à Z='c' par le biais de l'autre symbole.

2.14 Variables d'interprétation des nombres

Les variables d'interprétation des nombres sont définies par le système. Les variables sont incluses en haut du script de chargement et appliquent les paramètres de formatage des nombres au moment de l'exécution du script. Elles peuvent être supprimées, éditées ou dupliquées.

Les variables d'interprétation des nombres sont automatiquement générées en fonction des paramètres régionaux actifs du système d'exploitation à la création d'une application. Dans Qlik Sense Desktop, cela dépend des paramètres du système d'exploitation de l'ordinateur. Dans Qlik Sense, cela dépend du système d'exploitation du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Formatage de la devise

MoneyDecimalSep

Le séparateur décimal défini remplace le symbole décimal de la devise établi par vos paramètres régionaux.

```
MoneyDecimalSep
```

MoneyFormat

Le symbole défini remplace le symbole de la devise établi par vos paramètres régionaux.

```
MoneyFormat
```

MoneyThousandSep

Le séparateur des milliers défini remplace le symbole de groupement des chiffres de la devise établi par vos paramètres régionaux.

```
MoneyThousandSep
```

Formatage des nombres

DecimalSep

Le séparateur décimal défini remplace le symbole décimal établi par vos paramètres régionaux.

```
DecimalSep
```

ThousandSep

Le séparateur de milliers défini remplace le symbole de groupement des chiffres du système d'exploitation (configuré dans les paramètres régionaux).

```
ThousandSep
```

NumericalAbbreviation

L'abréviation numérique définit l'abréviation à utiliser pour les préfixes d'échelle des nombres, par exemple M pour méga ou un million (10^6), et µ pour micro (10^{-6}).

```
NumericalAbbreviation
```

Formatage de l'heure

DateFormat

Cette variable d'environnement définit le format de date utilisé comme format par défaut dans l'application. Le format est utilisé pour interpréter et formater les dates. Si la variable n'est pas définie, le format de date des paramètres régionaux du système d'exploitation sera récupéré lors de l'exécution du script.

```
DateFormat
```

TimeFormat

Le format défini remplace le format de l'heure du système d'exploitation (configuré dans les paramètres régionaux).

```
TimeFormat
```

TimestampFormat

Le format défini remplace les formats de date et heure du système d'exploitation (configurés dans les paramètres régionaux).

```
TimestampFormat
```

MonthNames

Le format défini remplace la convention de dénomination des mois des paramètres régionaux.

```
MonthNames
```

LongMonthNames

Le format défini remplace la convention de dénomination des mois longs des paramètres régionaux.

```
LongMonthNames
```

DayNames

Le format défini remplace la convention de dénomination des jours de la semaine établie par vos paramètres régionaux.

```
DayNames
```

LongDayNames

Le format défini remplace la convention de dénomination des jours de la semaine longs des paramètres régionaux.

```
LongDayNames
```

FirstWeekDay

Nombre entier qui définit le jour à utiliser comme premier jour de la semaine.

```
FirstWeekDay
```

BrokenWeeks

Ce paramètre définit si les semaines sont interrompues ou non.

```
BrokenWeeks
```

ReferenceDay

Le paramètre spécifie le jour du mois de janvier devant être défini comme jour de référence pour définir la semaine 1.

```
ReferenceDay
```

FirstMonthOfYear

Le paramètre définit le mois à utiliser comme premier mois de l'année. Il permet ainsi de spécifier les années financières faisant appel à un décalage mensuel (démarrage au 1er avril, par exemple).

Ce paramètre n'est pas utilisé pour le moment, mais il est réservé à un usage ultérieur.

Les paramètres valides sont compris entre 1 (janvier) et 12 (décembre). Le paramètre par défaut est 1.

Syntaxe :

```
FirstMonthOfYear
```

Exemple :

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

BrokenWeeks

Ce paramètre définit si les semaines sont interrompues ou non.

Syntaxe :

```
BrokenWeeks
```

Par défaut, les fonctions Qlik Sense utilisent des semaines ininterrompues. Autrement dit :

- Certaines années, la semaine 1 commence en décembre, tandis que d'autres années, la semaine 52 ou 53 se poursuit en janvier.
- La semaine 1 comporte toujours au moins 4 jours en janvier.

L'alternative consiste à utiliser des semaines interrompues :

- La semaine 52 ou 53 ne se poursuit pas en janvier.
- La semaine 1 débute le 1er janvier et n'est, dans la plupart des cas, pas une semaine complète.

Les valeurs suivantes peuvent être utilisées :

- 0 (= utilisation de semaines ininterrompues)
- 1 (= utilisation de semaines interrompues)

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples :

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

DateFormat

Cette variable d'environnement définit le format de date utilisé par défaut dans l'application et par date renvoyant des fonctions comme `date()` et `date#()`. Le format est utilisé pour interpréter et formater les dates. Si la variable n'est pas définie, le format de date défini par vos paramètres régionaux est récupéré lors de l'exécution du script.

Syntaxe :

```
DateFormat
```

Exemples de fonction DateFormat

Exemple	Résultat
`Set DateFormat='M/D/YY'; //(US format)`	Cette utilisation de la fonction `DateFormat` définit la date au format américain, à savoir, mois/jour/année.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Cette utilisation de la fonction `DateFormat` définit la date au format britannique, à savoir, jour/mois/année.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Cette utilisation de la fonction `DateFormat` définit la date au format ISO, à savoir, année/mois/jour.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – Variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates.
- La fonction `DateFormat`, qui utilisera le format de date américain.

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Transactions`'. Il inclut un champ `date`. La définition `DateFormat` américaine est utilisée. Ce modèle sera utilisé pour la conversion de texte en date implicite lorsque les dates textuelles sont chargées.

Script de chargement

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

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

```
04/01/2022,4,2431
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `month`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

La définition `DateFormat` MM/DD/YYYY est utilisée pour la conversion implicite de texte en dates. C'est pourquoi le champ `date` est correctement interprété comme une date. Le même format est utilisé pour afficher la date, comme indiqué dans le tableau de résultats.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans l'exemple précédent.
- La fonction `DateFormat`, qui utilisera le format 'DD/MM/YYYY'.

Script de chargement

```
SET DateFormat='DD/MM/YYYY';
Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `month`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

Étant donné que la définition `DateFormat` a été déterminée sur 'DD/MM/YYYY', vous pouvez voir que les deux chiffres après le premier symbole "/" ont été interprétés comme le mois, ce qui fait que tous les enregistrements proviennent du mois de janvier.

Exemple 3 – interprétation de la date

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données avec des dates au format numérique.
- La variable `DateFormat`, qui utilisera le format 'DD-MM-YYYY'.
- La variable `date()`.

Script de chargement

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

Transactions:
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `month`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

date	month	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

Dans le script de chargement, vous utilisez la fonction `date()` pour convertir la date numérique en un format de date. Étant donné que vous ne fournissez pas de format spécifié comme deuxième argument dans la fonction, le format `DateFormat` est utilisé. C'est pourquoi le champ de date utilise le format 'MM/DD/YYYY'.

Exemple 4 – formatage de dates étrangères

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates.
- La variable `DateFormat`, qui utilise le format 'DD/MM/YYYY', mais qui n'est pas commentée par des barres obliques.

Script de chargement

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


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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `month`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

Dans le script de chargement initial, le format `DateFormat` utilisé est le format 'MM/DD/YYYY' par défaut. Étant donné que le champ `date` de l'ensemble de données Transactions ne se présente dans ce format, le champ n'est pas interprété comme une date. Cela apparaît dans le tableau de résultats où les valeurs du champ `month` sont nulles.

Vous pouvez vérifier les types de données interprétés dans le visionneur de modèle de données en inspectant les propriétés "Tags" du champ `date` :

Aperçu du tableau `Transactions`. Notez les "Tags" du champ `date` indiquant que les données d'entrée textuelles n'ont pas été implicitement converties en date/horodatage.

date	
Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$ascii $text

Transactions

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

Cela peut être résolu en activant la variable système `DateFormat` :

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

Supprimez les doubles barres obliques et actualisez les données.

Aperçu du tableau `Transactions`. Notez les "Tags" du champ `date` indiquant que les données d'entrée textuelles ont été implicitement converties en date/horodatage.

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

Transactions

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

DayNames

Le format défini remplace la convention de dénomination des jours de la semaine établie par vos paramètres régionaux.

Syntaxe :

```
DayNames
```

Lors de la modification de la variable, un point-virgule ; est nécessaire pour séparer les valeurs individuelles.

Exemples de fonction DayName

Exemple de fonction	Définition du résultat
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Cette utilisation de la fonction DayNames définit les noms de jour sous leur forme abrégée.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Cette utilisation de la fonction DayNames définit les noms de jour par leurs premières lettres.

La fonction `DayNames` est souvent utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
weekday (page 1074)	Fonction de script permettant de renvoyer `DayNames` sous forme de valeurs de champ.
Date (page 1232)	Fonction de script permettant de renvoyer `DayNames` sous forme de valeurs de champ.
LongDayNames (page 225)	Valeurs au format long de `DayNames`.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, les dates de l'ensemble de données sont définies au format MM/DD/YYYY.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données avec des dates, qui sera chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `DayNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `dayname`

Créez cette mesure :

```
sum(amount)
```

Tableau de résultats

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

Dans le script de chargement, la fonction `WeekDay` est utilisée avec le champ `date` comme argument fourni. Dans le tableau de résultats, la sortie de cette fonction `WeekDay` affiche les jours de la semaine au format de la définition `DayNames`.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet. Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, au début du script, la définition `DayNames` est modifiée de sorte à utiliser les jours abrégés de la semaine en afrikaans.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `dayname`

Créez cette mesure :

```
sum(amount)
```

Tableau de résultats

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

Dans le tableau de résultats, la sortie de cette fonction `WeekDay` affiche les jours de la semaine au format de la définition `DayNames`.

Il est important de garder à l'esprit que si la langue de `DayNames` est modifiée comme dans cet exemple, la valeur `LongDayNames` contiendra toujours les jours de la semaine en anglais. Cela devrait également être modifié si les deux variables sont utilisées dans l'application.

Exemple 3 – fonction date

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données avec des dates, qui sera chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `DayNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `dayname`

Créez cette mesure :

```
sum(amount)
```

Tableau de résultats

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

La définition `DayNames` par défaut est utilisée. Dans le script de chargement, la fonction `Date` est utilisée avec le champ `date` comme premier argument. Le deuxième argument est `WWW`. Ce formatage convertit le résultat en valeurs stockées dans la définition `DayNames`. Cela apparaît dans la sortie du tableau de résultats.

DecimalSep

Le séparateur décimal défini remplace le symbole décimal établi par vos paramètres régionaux.

Qlik Sense interprète automatiquement le texte comme des nombres chaque fois qu'un modèle de nombre reconnaissable est rencontré. Les variables système `ThousandSep` et `DecimalSep` déterminent la composition des modèles appliqués lors de l'analyse du texte sous forme de nombres. Les variables `ThousandSep` et `DecimalSep` définissent le modèle de format numérique par défaut lors de la visualisation du contenu numérique dans des graphiques et tableaux frontaux. Autrement dit, cela a un impact direct sur les options **Formatage des nombres** pour toute expression frontale.

En supposant un séparateur de milliers de type virgule ',' et un séparateur de valeurs décimales de type point '.', voici des exemples de modèles qui seraient implicitement convertis en valeurs numériques équivalentes :

```
0,000.00
```

```
0000.00
```

```
0,000
```

Il s'agit d'exemples de modèles qui resteraient inchangés sous forme de texte ; c'est-à-dire, non convertis en valeurs numériques :

```
0.000,00
```

```
0,00
```

Syntaxe :

```
DecimalSep
```

Exemples de fonction

Exemple	Résultat
`Set DecimalSep='.';`	Définit '.' comme séparateur de valeurs décimales.

Exemple	Résultat
`Set DecimalSep=',';`	Définit ',' comme séparateur de valeurs décimales.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple - effet de la définition de variables de séparateur numérique sur différentes données d'entrée

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de sommes et de dates avec les sommes définies dans différents modèles de format.
- Table nommée `Transactions`.
- Variable `DecimalSep` définie sur '.'.
- Variable `ThousandSep` définie sur ','.
- Variable `delimiter` définie comme caractère '|' pour séparer les différents champs sur une ligne.

Script de chargement

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

Transactions:
Load date,
id,
amount as amount
Inline
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`amount`.

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

Montant	=Sum(amount)	
Totals		20814.7086
1.000-45		
3.756,178		
4124,35		
	23.344	23.344
	2431.36	2431.36
	2431.84	2431.84
	3,454.356	3454.356
	3554.284	3554.284
	4132.5246	4132.5246
	4,787	4787

Toute valeur non interprétée comme un nombre reste au format texte et est alignée à gauche par défaut. Toutes les valeurs correctement converties sont alignées à droite, en conservant le format d'entrée d'origine.

La colonne d'expression affiche l'équivalent numérique, qui est formaté par défaut avec uniquement un séparateur décimal '.'. Cela peut être remplacé par le paramètre déroulant **Formatage des nombres** dans la configuration de l'expression.

FirstWeekDay

Nombre entier qui définit le jour à utiliser comme premier jour de la semaine.

Syntaxe :

```
FirstWeekDay
```

Par défaut, les variables système Qlik Sense définissent `FirstWeekDay=6`. Cela signifie que Dimanche est le premier jour de la semaine.

Valeurs pouvant être définies pour FirstWeekDay

Valeur	Jour
0	Lundi
1	Mardi
2	Mercredi
3	Jeudi
4	Vendredi
5	Samedi
6	Dimanche

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – Utilisation de la valeur par défaut (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Dans cet exemple, le script de chargement utilise la valeur de la variable système Qlik Sense par défaut, `FirstWeekDay=6`. Ces données contiennent des données pour les 14 premiers jours de 2020.

Script de chargement

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

SET FirstWeekDay = 6;

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau des résultats

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

Étant donné que les paramètres par défaut sont utilisés, la variable système `FirstWeekDay` est définie sur 6. Dans le tableau de résultats, chaque nouvelle semaine apparaît comme commençant le dimanche (le 5 et le 12 du mois de janvier).

Exemple 2 – Modification de la variable FirstWeekDay (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Dans cet exemple, les données contiennent les 14 premiers jours de 2020. Au début du script, nous définissons la variable `FirstWeekDay` sur 3.

Script de chargement

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

SET FirstWeekDay = 3;

Sales:
LOAD
       date,
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau des résultats

Date	semaine	weekday
01/01/2021	52	Wed
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	1	Sun
01/06/2020	1	Mon
01/07/2020	1	Tue
01/08/2020	1	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	2	Sun

Date	semaine	weekday
01/13/2020	2	Mon
01/14/2020	2	Tue

Étant donné que la variable système `FirstWeekDay` est définie sur 3, le premier jour de chaque semaine sera un jeudi. Dans le tableau de résultats, chaque nouvelle semaine apparaît comme commençant le jeudi (le 2 et le 9 du mois de janvier).

LongDayNames

Le format défini remplace la convention de dénomination des jours de la semaine longs des paramètres régionaux.

Syntaxe :

```
LongDayNames
```

L'exemple suivant de la fonction `LongDayNames` définit les noms de jour au format complet :

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

Lors de la modification de la variable, un point-virgule ; est nécessaire pour séparer les valeurs individuelles.

La fonction `LongDayNames` peut être utilisée en combinaison avec la fonction *Date (page 1232)* qui renvoie `DayNames` comme valeurs de champ.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – variable système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données avec des dates, qui sera chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `LongDayNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `dayname`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

date	dayname	=sum(amount)
01/01/2022	Samedi	1000
02/01/2022	Mardi	2123
03/01/2022	Mardi	4124
04/01/2022	Vendredi	2431

Dans le script de chargement, pour créer un champ nommé `dayname`, la fonction `Date` est utilisée avec le champ `date` comme premier argument. Le deuxième argument de la fonction est le formatage `WWWW`.

L'utilisation de ce formatage convertit les valeurs du premier argument en nom de jour complet correspondant défini dans la variable `LongDayNames`. Dans le tableau de résultats, les valeurs de champ de notre champ créé `dayname` affichent ceci.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés. Cependant, au début du script, la définition `LongDayNames` est modifiée de sorte à utiliser les jours de la semaine en espagnol.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `dayname`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

Dans le script de chargement, la variable LongDayNames est modifiée de sorte à répertorier les jours de la semaine en espagnol.

Créez ensuite un champ, nommé dayname, qui est la fonction Date utilisée avec le champ date comme premier argument.

Le deuxième argument de la fonction est le formatage .WWWW L'utilisation de ce formatage Qlik Sense convertit les valeurs du premier argument en nom de jour complet correspondant défini dans la variable LongDayNames.

Dans le tableau de résultats, les valeurs de champ de notre champ créé dayname affichent les jours de la semaine écrits en espagnol et en entier.

LongMonthNames

Le format défini remplace la convention de dénomination des mois longs des paramètres régionaux.

Syntaxe :

```
LongMonthNames
```

Lors de la modification de la variable, un point-virgule ; est nécessaire pour séparer les valeurs individuelles.

L'exemple suivant de la fonction LongMonthNames définit les noms de mois au format complet :

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

La fonction LongMonthNames est souvent utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
Date (page 1232)	Fonction de script permettant de renvoyer DayNames sous forme de valeurs de champ.
LongDayNames (page 225)	Valeurs au format long de DayNames.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction SET DateFormat de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, qui est chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `LongMonthNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- monthname

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

La définition `LongMonthNames` par défaut est utilisée. Dans le script de chargement, pour créer un champ nommé `month`, la fonction `Date` est utilisée avec le champ `date` comme premier argument. Le deuxième argument de la fonction est le formatage .MMMM

L'utilisation de ce formatage Qlik Sense convertit les valeurs du premier argument en nom de mois complet correspondant défini dans la variable `LongMonthNames`. Dans le tableau de résultats, les valeurs de champ de notre champ créé `month` affichent ceci.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, qui est chargé dans une table nommée `Transactions`.
- Champ `date`.
- Variable `LongMonthNames` modifiée de sorte à utiliser les jours abrégés de la semaine en espagnol.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez `sum(amount)` comme mesure et ces champs comme dimensions :

- `date`
- `monthname`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

Dans le script de chargement, la variable `LongMonthNames` est modifiée de sorte à répertorier les mois de l'année en espagnol. Ensuite, pour créer un champ nommé `monthname`, la fonction `Date` est utilisée avec le champ `date` comme premier argument. Le deuxième argument de la fonction est le formatage .MMMM

L'utilisation de ce formatage Qlik Sense convertit les valeurs du premier argument en nom de mois complet correspondant défini dans la variable `LongMonthNames`. Dans le tableau de résultats, les valeurs de champ de notre champ créé `monthname` affichent le nom du mois écrit en espagnol.

MoneyDecimalSep

Le séparateur décimal défini remplace le symbole décimal de la devise établi par vos paramètres régionaux.

Par défaut, Qlik Sense affiche les nombres et le texte différemment dans les graphiques de tableau. Les nombres sont alignés à droite tandis que le texte est aligné à gauche. Cela facilite la recherche des problèmes de conversion de texte en nombre. Tous les tableaux de cette page qui affichent des résultats Qlik Sense utiliseront cette mise en page.

Syntaxe :

```
MoneyDecimalSep
```

Les applications Qlik Sense interpréteront les champs de texte conformes à ce formatage comme des valeurs monétaires. Le champ de texte doit contenir le symbole de devise défini dans la variable système `MoneyFormat`. `MoneyDecimalSep` s'avère particulièrement utile lors du traitement de sources de données reçues provenant de plusieurs paramètres régionaux différents.

L'exemple suivant montre une utilisation possible de la variable système `MoneyDecimalSep` :

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

Cette fonction est souvent utilisée avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
`MoneyFormat`	Dans les instances d'interprétation de champs de texte, le symbole `MoneyFormat` sera utilisé dans le cadre de l'interprétation. Pour Formatage des nombres, le formatage `MoneyFormat` sera utilisé par Qlik Sense dans les objets graphiques.
`MoneyThousandSep`	Dans les instances d'interprétation de champs de texte, la fonction `MoneyThousandSep` doit également être suivie.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - Notation point (.) MoneyDecimalSep

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Données fournies avec leur champ monétaire au format texte utilisant un point '.' comme séparateur décimal. Chaque enregistrement est également préfixé d'un symbole '$', sauf le dernier enregistrement, qui est préfixé d'un symbole '£'.

N'oubliez pas que la variable système `MoneyFormat` définit le dollar '$' comme la devise par défaut.

Script de chargement

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

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

```
01/10/2022,10,'£345.66'
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`amount`.

Ajoutez les mesures suivantes :

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

Examinez les résultats ci-dessous, qui montrent l'interprétation correcte de toutes les valeurs dollar '$' uniquement.

Tableau de résultats

amount	=isNum(amount)	=Sum(amount)
Totals	0	$3905.98
£345.66	0	$0.00
$3.436	-1	$3.44
$3.55	-1	$3.55
$7.54	-1	$7.54
$14.41	-1	$14.41
$24.37	-1	$24.37
243.63	-1	$243.63
$249.36	-1	$249.36
$545.36	-1	$545.36
$2,814.32	-1	$2814.32

Le tableau de résultats ci-dessus montre comment le champ `amount` a été interprété correctement pour toutes les valeurs préfixées du symbole dollar ($), tandis que la valeur `amount`préfixée du symbole livre (£) n'a pas été convertie en valeur monétaire.

Exemple 2 - Notation virgule (,) MoneyDecimalSep

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Données fournies avec leur champ monétaire au format texte utilisant une virgule ‘,’ comme séparateur décimal. Chaque enregistrement est également préfixé d'un symbole ‘$’, sauf le dernier enregistrement, qui utilise par erreur le séparateur décimal point '.'.

N'oubliez pas que la variable système `MoneyFormat` définit le dollar ‘$’ comme la devise par défaut.

Script de chargement

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

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14,41'
01/02/2022,2,'$2.814,32'
01/03/2022,3,'$249,36'
01/04/2022,4,'$24,37'
01/05/2022,5,'$7,54'
01/06/2022,6,'$243,63'
01/07/2022,7,'$545,36'
01/08/2022,8,'$3,55'
01/09/2022,9,'$3,436'
01/10/2022,10,'$345.66'
];
```

Résultats

Texte de paragraphe des résultats.

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`amount`.

Ajoutez les mesures suivantes :

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

Examinez les résultats ci-dessous, montrant l'interprétation correcte de l'ensemble des valeurs, sauf le montant pour lequel le séparateur décimal utilise une notation point '.'. Dans chaque cas, une virgule aurait dû être utilisée à la place.

Tableau de résultats

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

MoneyFormat

Cette variable système définit le modèle de format utilisé par Qlik pour la traduction automatique de texte en nombre lorsque le nombre est préfixé d'un symbole monétaire. Elle détermine également la manière dont les mesures dont les propriétés Formatage de nombres sont définies sur 'Devise' seront affichées dans les objets graphiques.

Le symbole défini dans le cadre du modèle de format de la variable système `MoneyFormat` remplace le symbole de devise établi par vos paramètres régionaux.

Par défaut, Qlik Sense affiche les nombres et le texte différemment dans les graphiques de tableau. Les nombres sont alignés à droite tandis que le texte est aligné à gauche. Cela facilite la recherche des problèmes de conversion de texte en nombre. Tous les tableaux de cette page qui affichent des résultats Qlik Sense utiliseront cette mise en page.

Syntaxe :

```
MoneyFormat
```

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

Ce formatage sera affiché dans les objets graphiques lorsque la propriété `Number Formatting` d'un champ numérique est définie sur `Money`. De plus, lorsque des champs de texte numériques sont interprétés par Qlik Sense, si le symbole de devise du champ de texte correspond au symbole défini dans la variable `MoneyFormat`, Qlik Sense interprétera ce champ comme une valeur monétaire.

Cette fonction est souvent utilisée avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
MoneyDecimalSep (page 232)	Pour Formatage des nombres, `MoneyDecimalSep` sera utilisé dans le formatage de champ des objets.
MoneyThousandSep (page 240)	Pour Formatage des nombres, `MoneyThousandSep` sera utilisé dans le formatage de champ des objets.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - MoneyFormat

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient un ensemble de données qui est chargé dans une table nommée `Transactions`. La définition de la variable `MoneyFormat` par défaut est utilisée.

Script de chargement

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

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `amount`

Ajoutez cette mesure :

```
=Sum(amount)
```

Sous **Formatage de nombres**, sélectionnez **Devise** pour configurer `Sum(amount)` comme une valeur monétaire.

Tableau de résultats

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

La définition `MoneyFormat` par défaut est utilisée. Cela prend la forme suivante : `$###0.00;-$###0.00`. Dans le tableau de résultats, le format du champ `amount` affiche le symbole de devise et le point décimal et les valeurs décimales ont été incluses.

Exemple 2 - MoneyFormat avec séparateur de milliers et formats d'entrée mixtes

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données aux formats d'entrée mixtes, chargé dans une table appelée `Transactions` avec des séparateurs de milliers et des séparateurs décimaux parsemés.
- Une modification de la définition `MoneyFormat` inclut une virgule comme séparateur de milliers.
- Une des lignes de données est délimitée par erreur avec des séparateurs de milliers de type virgule aux mauvais endroits. Notez comment ce montant est laissé sous forme de texte sans pouvoir être interprété comme un nombre.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `amount`

Ajoutez cette mesure :

```
=Sum(amount)
```

Sous **Formatage de nombres**, sélectionnez **Devise** pour configurer `Sum(amount)` comme une valeur monétaire.

Tableau de résultats

date	Montant	=Sum(amount)
Totals		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10,000,000,441.45
01/02/2022	$212,3749,24,32.23	$0.00
01/03/2022	$249475336.45	$249,475,336.45
01/04/2022	$24	$24.00
01/05/2022	$7873578754	$7,873,578,754.00
01/06/2022	$24313884663	$24,313,884,663.00
01/07/2022	$545883436	$545,883,436.00
01/08/2022	$35545828255	$35,545,828,255.00
01/09/2022	$37565817436	$37,565,817,436.00
01/10/2022	$3454343566	$3,454,343,566.00

Au début du script, la variable système `MoneyFormat` est modifiée de sorte à inclure une virgule comme séparateur de milliers. Dans la table Qlik Sense, le formatage apparaît de sorte à inclure ce séparateur. De plus, la ligne contenant le séparateur erroné n'a pas été correctement interprétée et reste au format texte. C'est pourquoi elle ne contribue pas à la somme du montant.

MoneyThousandSep

Le séparateur des milliers défini remplace le symbole de groupement des chiffres de la devise établi par vos paramètres régionaux.

Par défaut, Qlik Sense affiche les nombres et le texte différemment dans les graphiques de tableau. Les nombres sont alignés à droite tandis que le texte est aligné à gauche. Cela facilite la recherche des problèmes de conversion de texte en nombre. Tous les tableaux de cette page qui affichent des résultats Qlik Sense utiliseront cette mise en page.

Syntaxe :

```
MoneyThousandSep
```

Les applications Qlik Sense interpréteront les champs de texte conformes à ce formatage comme des valeurs monétaires. Le champ de texte doit contenir le symbole de devise défini dans la variable système `MoneyFormat`. `MoneyThousandSep` s'avère particulièrement utile lors du traitement de sources de données reçues provenant de plusieurs paramètres régionaux différents.

L'exemple suivant montre une utilisation possible de la variable système `MoneyThousandSep` :

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

Cette fonction est souvent utilisée avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
`MoneyFormat`	Dans les instances d'interprétation de champs de texte, le symbole `MoneyFormat` sera utilisé dans le cadre de l'interprétation. Pour Formatage des nombres, le formatage `MoneyFormat` sera utilisé par Qlik Sense dans les objets graphiques.
`MoneyDecimalSep`	Dans les instances d'interprétation de champs de texte, la fonction `MoneyDecimalSep` doit également être suivie.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - Notation virgule (,) MoneyThousandSep

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Données fournies avec leur champ monétaire au format texte utilisant une virgule ‘,’ comme séparateur de milliers. Chaque enregistrement est également préfixé d'un symbole ‘$’.

N'oubliez pas que la variable système `MoneyFormat` définit le dollar ‘$’ comme la devise par défaut.

Script de chargement

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

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441'
01/02/2022,2,'$21,237,492,432'
01/03/2022,3,'$249,475,336'
01/04/2022,4,'$24,313,369,837'
01/05/2022,5,'$7,873,578,754'
01/06/2022,6,'$24,313,884,663'
01/07/2022,7,'$545,883,436'
01/08/2022,8,'$35,545,828,255'
01/09/2022,9,'$37,565,817,436'
01/10/2022,10,'$3.454.343.566'
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`amount`.

Ajoutez les mesures suivantes :

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

Examinez les résultats ci-dessous. Le tableau montre l'interprétation correcte de l'ensemble des valeurs utilisant la notation virgule ‘,’ comme séparateur de milliers.

Le champ `amount` a été correctement interprété pour toutes les valeurs, sauf une, qui utilisait un point '.' comme séparateur de milliers.

Tableau de résultats

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

Exemple 2 - Notation point (.) MoneyThousandSep

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Données fournies avec leur champ monétaire au format texte utilisant un point '.' comme séparateur de milliers. Chaque enregistrement est également préfixé d'un symbole '$'.

N'oubliez pas que la variable système `MoneyFormat` définit le dollar '$' comme la devise par défaut.

Script de chargement

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

Transactions:
Load
date,
id,
amount
Inline
[
```

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`amount`.

Ajoutez les mesures suivantes :

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

Examinez les résultats ci-dessous, montrant l'interprétation correcte de l'ensemble des valeurs utilisant la notation point '.' comme séparateur de milliers.

Le champ `amount` a été correctement interprété pour toutes les valeurs, sauf une, qui utilisait une virgule ',' comme séparateur de milliers.

Tableau de résultats

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

MonthNames

Le format défini remplace la convention de dénomination des mois des paramètres régionaux.

Syntaxe :

```
MonthNames
```

Lors de la modification de la variable, un point-virgule ; est nécessaire pour séparer les valeurs individuelles.

Exemples de fonction

Exemple	Résultats
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Cette utilisation de la fonction `MonthNames` définit les noms de mois en anglais et leur forme abrégée.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Cette utilisation de la fonction `MonthNames` définit les noms de mois en espagnol et leur forme abrégée.

La fonction `MonthNames` peut être utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
month (page 912)	Fonction de script permettant de renvoyer des valeurs définies dans `MonthNames` sous forme de valeurs de champ
Date (page 1232)	Fonction de script permettant de renvoyer des valeurs définies dans `MonthNames` sous forme de valeurs de champ basées sur un argument de formatage fourni
LongMonthNames (page 228)	Valeurs au format long de `MonthNames`

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – Variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, qui est chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `MonthNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `monthname`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

La définition `MonthNames` par défaut est utilisée. Dans le script de chargement, la fonction `Month` est utilisée avec le champ `date` comme argument fourni.

Dans le tableau de résultats, la sortie de cette fonction `Month` affiche les mois de l'année au format de la définition `MonthNames`.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, qui est chargé dans une table nommée `Transactions`.
- Champ `date`.
- Variable `MonthNames` modifiée de sorte à utiliser les mois abrégés en espagnol.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `monthname`

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

Dans le script de chargement, pour commencer, la variable `MonthNames` est modifiée de sorte à répertorier les mois de l'année abrégés en espagnol. La fonction `Month` est utilisée avec le champ `date` comme argument fourni.

Dans le tableau de résultats, la sortie de cette fonction `Month` affiche les mois de l'année au format de la définition `MonthNames`.

Il est important de garder à l'esprit que si la langue de la variable `MonthNames` est modifiée, comme dans cet exemple, la variable `LongMonthNames` contiendra toujours les mois de l'année en anglais. La variable `LongMonthNames` devrait être modifié si les deux variables sont utilisées dans l'application.

Exemple 3 – fonction date

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, qui est chargé dans une table nommée `Transactions`.
- Champ `date`.
- Définition `MonthNames` par défaut.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- monthname

Créez cette mesure :

```
=sum(amount)
```

Tableau de résultats

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

La définition `MonthNames` par défaut est utilisée. Dans le script de chargement, la fonction `Date` est utilisée avec le champ `date` comme premier argument. Le deuxième argument est .MMM

L'utilisation de ce formatage Qlik Sense convertit les valeurs du premier argument en nom de mois correspondant défini dans la variable `MonthNames`. Dans le tableau de résultats, les valeurs de champ de notre champ créé `month` affichent ceci.

NumericalAbbreviation

L'abréviation numérique définit l'abréviation à utiliser pour les préfixes d'échelle des nombres, par exemple M pour méga ou un million (10^6), et µ pour micro (10^{-6}).

Syntaxe :

```
NumericalAbbreviation
```

Vous définissez la variable `NumericalAbbreviation` sur une chaîne contenant une liste de paires de définitions d'abréviations, délimitées par des points-virgules. Chaque paire de définitions d'abréviations doit contenir l'échelle (l'exposant en base décimale) et l'abréviation séparées par deux-points, par exemple `6:M` pour un million.

Le paramètre par défaut est `'3:k;6:M;9:G;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y'`.

Exemples :

Ce paramètre remplace le préfixe d'un millier par t et le préfixe d'un milliard par B. Cette méthode peut s'avérer pratique pour des applications financières dans lesquelles des abréviations de type t$, M$ et B$ sont courantes.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

Le paramètre définit le jour de janvier à déterminer comme jour de référence pour définir la semaine 1. En d'autres termes, ce paramètre prescrit le nombre de jours de la semaine 1 qui doivent être des dates en janvier.

Syntaxe :

```
ReferenceDay
```

`ReferenceDay` définit le nombre de jours inclus dans la première semaine de l'année. `ReferenceDay` peut être réglé sur n'importe quelle valeur comprise entre `1` et `7`. Toute valeur en dehors de la plage `1-7` est interprétée comme le milieu de la semaine (`4`), ce qui équivaut à définir `ReferenceDay` sur `4`.

Si vous ne sélectionnez pas de valeur pour le paramètre `ReferenceDay`, la valeur par défaut affichera `ReferenceDay=0`, qui sera interprété comme le milieu de la semaine (`4`), comme indiqué dans le tableau de valeurs `ReferenceDay` ci-dessous.

La fonction `ReferenceDay` est souvent utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Variable	Interaction
BrokenWeeks (page 206)	Si l'application Qlik Sense fonctionne avec des semaines ininterrompues, le paramètre de variable `ReferenceDay` sera appliqué. Cependant, si des semaines interrompues sont utilisées, la semaine 1 commencera le 1er janvier et se terminera en conjonction avec le paramètre de variable `FirstWeekDay` et ignorera l'indicateur `ReferenceDay`.
FirstWeekDay (page 221)	Nombre entier qui définit le jour à utiliser comme premier jour de la semaine.

Qlik Sense permet de définir les valeurs suivantes pour `ReferenceDay` :

Valeurs ReferenceDay

Valeur	Jour de référence
0 (par défaut)	4 janvier
1	1er janvier
2	2 janvier

Valeur	Jour de référence
3	3 janvier
4	4 janvier
5	5 janvier
6	6 janvier
7	7 janvier

Dans l'exemple suivant, la valeur `ReferenceDay = 3` définit le 3 janvier comme jour de référence :

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

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – script de chargement utilisant la valeur par défaut ; ReferenceDay=0

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Variable `ReferenceDay` définie sur `0`.
- Variable `BrokenWeeks` définie sur `0` qui force l'application à utiliser des semaines ininterrompues.
- Ensemble de données de dates de la fin de 2019 au début de 2020.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau de résultats

date	semaine	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	1	Sun
12/30/2019	1	Mon
12/31/2019	1	Tue
01/01/2020	1	Wed
01/02/2020	1	Thu
01/03/2020	1	Fri
01/04/2020	1	Sat
01/05/2020	2	Sun

date	semaine	weekday
01/06/2020	2	Mon
01/07/2020	2	Tue
01/08/2020	2	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat

La semaine 52 se termine le samedi 28 décembre. Étant donné que `ReferenceDay` exige que le 4 janvier soit inclus dans la semaine 1, la semaine 1 commence le 29 décembre et se termine le samedi 4 janvier.

Exemple - variable ReferenceDay définie sur 5

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Variable `ReferenceDay` définie sur `5`.
- Variable `BrokenWeeks` définie sur `0` qui force l'application à utiliser des semaines ininterrompues.
- Ensemble de données de dates de la fin de 2019 au début de 2020.

Script de chargement

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

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
```

```
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

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

Tableau de résultats

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

La semaine 52 se termine le samedi 28 décembre. La variable `BrokenWeeks` force l'application à utiliser des semaines ininterrompues. La valeur de jour de référence de `5` exige que le 5 janvier soit inclus dans la semaine 1.

Cependant, cela tombe huit jours après la fin de la semaine 52 de l'année précédente. Par conséquent, la semaine 53 commence le 29 décembre et se termine le 4 janvier. La semaine 1 commence le dimanche 5 janvier.

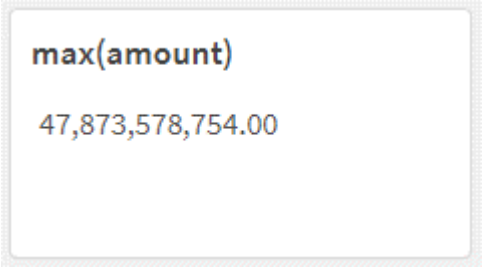
ThousandSep

Le séparateur de milliers défini remplace le symbole de groupement des chiffres du système d'exploitation (configuré dans les paramètres régionaux).

Syntaxe :

ThousandSep

Objet Qlik Sense utilisant la variable ThousandSep (avec séparateur de milliers)



Les applications Qlik Sense interprètent les champs de texte conformes à ce formatage comme des nombres. Ce formatage sera affiché dans les objets graphiques lorsque la propriété **Formatage de nombres** d'un champ numérique est définie sur **Nombre**.

ThousandSep s'avère utile lors du traitement de sources de données reçues de différents paramètres régionaux.

> *Si la variable ThousandSep est modifiée après la création et le formatage des objets dans l'application, l'utilisateur devra reformater chaque champ pertinent en désélectionnant, puis en resélectionnant la propriété **Formatage de nombres Nombre**.*

Les exemples suivants montrent des utilisations possibles de la variable système ThousandSep :

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

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

Ces rubriques peuvent vous aider à utiliser cette fonction :

Rubriques connexes

Rubrique	Description
DecimalSep (page 218)	Dans les instances d'interprétation de champs de texte, les paramètres de séparateur décimal tels que fournis par cette fonction doivent eux aussi être respectés. Pour Formatage des nombres, **DecimalSep** sera utilisé par Qlik Sense, le cas échéant.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction SET DateFormat de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Utilisation de la définition de la variable `ThousandSep` par défaut.

Script de chargement

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

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.
2. Ajoutez la mesure suivante :
   `=sum(amount)`
3. Dans le panneau des propriétés, sous **Données**, sélectionnez la mesure.
4. Sous **Formatage des nombres**, sélectionnez **Nombre**.

Ajustement du formatage des nombres pour une mesure de graphique

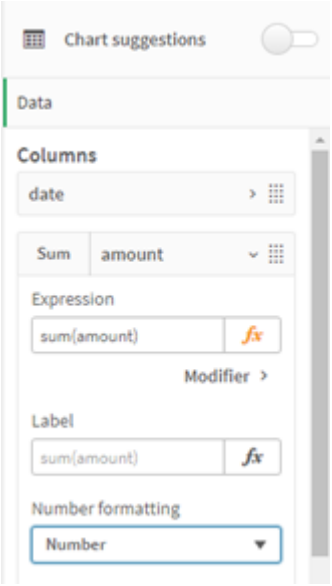


Tableau de résultats

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

Dans cet exemple, la définition `ThousandSep` par défaut sur le format virgule (‘,’) est utilisée. Dans le tableau de résultats, le format du champ amount affiche une virgule entre les regroupements de milliers.

Exemple 2 – modification de la variable système

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données que celui du premier exemple, chargé dans une table appelée `Transactions`.
- Modification de la définition `ThousandSep`, au début du script, de sorte à afficher un caractère '*' comme séparateur de milliers. Il s'agit d'un exemple extrême, utilisé uniquement pour démontrer la fonctionnalité de la variable.

La modification utilisée dans cet exemple est extrême et elle n'est pas couramment utilisée, mais, ici, elle a pour objectif de démontrer la fonctionnalité de la variable.

Script de chargement

```
SET ThousandSep='*';

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

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.
2. Ajoutez la mesure suivante :
   `=sum(amount)`
3. Dans le panneau des propriétés, sous **Données**, sélectionnez la mesure.
4. Sous **Formatage des nombres**, sélectionnez **Personnalisé**.

Tableau de résultats

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

Au début du script, la variable système `ThousandSep` est remplacée par un astérisque '*'. Dans le tableau de résultats, le format du champ amount affiche un '*' entre le regroupement de milliers.

Exemple 3 – interprétation du texte

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Transactions`.
- Données avec leur champ numérique au format texte utilisant une virgule comme séparateur de

milliers.

- Utilisation de la variable système `ThousandSep` par défaut.

Script de chargement

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

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.
2. Ajoutez la mesure suivante :
   `=sum(amount)`
3. Dans le panneau des propriétés, sous **Données**, sélectionnez la mesure.
4. Sous **Formatage des nombres**, sélectionnez **Nombre**.
5. Ajoutez la mesure suivante pour évaluer si le champ amount est ou non une valeur numérique :
   `=isnum(amount)`

Tableau de résultats

date	=sum(amount)	=isnum(amount)
01/01/2022	10,000,000,441.00	-1
01/02/2022	21,492,432.00	-1
01/03/2022	4,249,475,336.00	-1
01/04/2022	24,313,369,837.00	-1
01/05/2022	4,873,578,754.00	-1

date	=sum(amount)	=isnum(amount)
01/06/2022	313,884,663.00	-1
01/07/2022	2,545,883,436.00	-1
01/08/2022	545,828,255.00	-1
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

Une fois les données chargées, vous voyez que Qlik Sense a interprété que le champ amount comme une valeur numérique, en raison de la conformité des données à la variable `ThousandSep`. Cela est démontré par la fonction `isnum()`, qui évalue chaque entrée par rapport à -1 ou `TRUE`.

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

TimeFormat

Le format défini remplace le format de l'heure du système d'exploitation (configuré dans les paramètres régionaux).

Syntaxe :

```
TimeFormat
```

Exemple :

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

TimestampFormat

Le format défini remplace les formats de date et heure du système d'exploitation (configurés dans les paramètres régionaux).

Syntaxe :

```
TimestampFormat
```

Exemple :

Les exemples suivants utilisent *1983-12-14T13:15:30Z* comme données d'horodatage afin d'afficher les résultats de différentes instructions **SET TimestampFormat**. Le format de date utilisé est **YYYYMMDD**, tandis que le format de l'heure est **h:mm:ss TT**. Le format de date est indiqué dans l'instruction **SET DateFormat** et le format de l'heure dans l'instruction **SET TimeFormat**, situées en haut de votre script de chargement de données.

Résultats

Exemple	Résultat
`SET TimestampFormat='YYYYMMDD';`	19831214
`SET TimestampFormat='M/D/YY hh:mm:ss[.fff]';`	12/14/83 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';`	14/12/1983 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff] TT';`	14/12/1983 1:15:30 PM
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Exemples : Script de chargement

Exemple : Script de chargement

Dans le premier script de chargement, l'instruction *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'* est utilisée. Dans le deuxième script de chargement, le format d'horodatage est remplacé par *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. Ces résultats affichent la façon dont l'instruction **SET TimeFormat** fonctionne avec différents formats de données horaires.

La table ci-dessous affiche l'ensemble de données utilisé dans les scripts de chargement qui suivent. La deuxième colonne de la table indique le format de chaque horodatage dans l'ensemble de données. Les cinq premiers horodatages respectent les règles ISO 8601, contrairement au sixième.

Ensemble de données

Table affichant les données horaires utilisées et le format associé à chaque horodatage dans l'ensemble de données.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

Dans l'**éditeur de chargement de données**, créez une section, puis ajoutez et exécutez l'exemple de script. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de votre application afin de visualiser le résultat.

Script de chargement

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
```

```
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Résultats

Table Qlik Sense affichant les résultats de la variable d'interprétation TimestampFormat utilisée dans le script de chargement. Le dernier horodatage utilisé dans l'ensemble de données ne renvoie pas une date correcte.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Le script de chargement suivant utilise le même ensemble de données. Toutefois, il fait appel à *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* pour correspondre au format non-ISO 8601 du sixième horodatage.

Dans l'**éditeur de chargement de données**, remplacez l'exemple de script précédent par celui indiqué ci-dessous, puis exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de votre application afin de visualiser le résultat.

Script de chargement

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Résultats

Table Qlik Sense affichant les résultats de la variable d'interprétation TimestampFormat utilisée dans le script de chargement.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

2.15 Variables Direct Discovery

Variables système Direct Discovery

DirectCacheSeconds

Vous pouvez définir une limite de mise en cache applicable aux résultats de requêtes Direct Discovery pour les visualisations. Dès lors que cette limite temporelle est atteinte, Qlik Sense efface le contenu du cache à mesure que de nouvelles requêtes Direct Discovery sont effectuées. Qlik Sense recherche les sélections dans les données source et recrée le cache pour la durée limite spécifiée. Le résultat de chaque combinaison de sélections est mis en cache de manière indépendante. Autrement dit, le cache est actualisé pour chaque sélection de manière individuelle : une première sélection actualise le cache des seuls champs sélectionnés tandis qu'une seconde sélection actualise le cache des champs pertinents associés. Si la seconde sélection comprend des champs qui ont déjà été actualisés lors de la première sélection, ils ne sont pas remis à jour dans le cache si la limite de mise en cache n'a pas été atteinte.

Le cache Direct Discovery ne s'applique pas aux visualisations de type **Table**. Les sélections de table interrogent la source de données lors de chaque requête.

La valeur limite doit être définie en secondes. La limite de mise en cache par défaut est de 1 800 secondes (30 minutes).

La valeur utilisée pour la variable **DirectCacheSeconds** correspond à la valeur définie lors de l'exécution de l'instruction **DIRECT QUERY**. Il n'est pas possible de la modifier pendant que le programme est exécuté.

Exemple :

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Le regroupement de connexions vous permet d'émettre des appels parallèles asynchrones en direction de la base de données. La syntaxe du script de chargement permettant de configurer la fonction de regroupement est la suivante :

```
SET DirectConnectionMax=10;
```

Le paramètre numérique spécifie le nombre maximal de connexions à la base de données que le code Direct Discovery doit utiliser pendant la mise à jour d'une feuille. Le paramètre par défaut est 1.

> *Cette variable doit être utilisée avec précaution. En effet, si elle est définie sur une valeur supérieure à 1, elle provoque des problèmes lors de l'établissement de connexions avec Microsoft SQL Server.*

DirectUnicodeStrings

La fonction Direct Discovery prend en charge la sélection de données Unicode étendues via le format standard SQL pour les littéraux de chaîne de caractères étendue (N'<chaîne étendue>'), comme l'exigent certaines bases de données (tout particulièrement SQL Server). Vous pouvez activer l'utilisation de cette syntaxe pour Direct Discovery à l'aide de la variable de script **DirectUnicodeStrings**.

Si vous définissez cette variable sur 'true', vous pouvez faire précéder les littéraux de chaîne du marqueur "N" de caractère large de la norme ANSI. Toutes les bases de données ne prennent pas en charge cette norme. Le paramètre par défaut est 'false'.

DirectDistinctSupport

Lorsqu'une valeur de champ **DIMENSION** est sélectionnée dans un objet Qlik Sense, une requête est générée pour la base de données source. Lorsque la requête nécessite un regroupement, la fonction Direct Discovery fait appel au mot-clé **DISTINCT** pour ne sélectionner que des valeurs uniques. Certaines bases de données requièrent toutefois l'utilisation du mot-clé **GROUP BY**. Définissez la variable **DirectDistinctSupport** sur 'false' afin de générer le mot-clé **GROUP BY** au lieu du mot-clé **DISTINCT** dans les requêtes de valeurs uniques.

```
SET DirectDistinctSupport='false';
```

Si la variable DirectDistinctSupport est définie sur 'true', alors le mot-clé **DISTINCT** est utilisé. Si elle n'est pas définie, le comportement par défaut consiste à utiliser **DISTINCT**.

DirectEnableSubquery

Dans les scénarios à plusieurs tables à cardinalité élevée, il est possible de générer des sous-requêtes dans la requête SQL plutôt que de générer une longue clause IN. Pour ce faire, il est nécessaire de définir la variable **DirectEnableSubquery** sur 'true'. La valeur par défaut est 'false'.

> *Lorsque la variable **DirectEnableSubquery** est activée, vous ne pouvez pas charger de tables définies dans d'autres modes que le mode Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Variables de la fonction Bandes de requête de Teradata

La fonction Bandes de requête de Teradata permet aux applications d'entreprise de collaborer avec la base de données Teradata sous-jacente pour mieux gérer la comptabilité, la définition des priorités et la charge de travail. La fonction Bandes de requête vous permet d'englober une requête dans des métadonnées (des informations d'identification utilisateur, par exemple).

Deux variables sont disponibles. Toutes deux sont des chaînes qui sont évaluées et envoyées à la base de données.

SQLSessionPrefix
Cette chaîne est envoyée suite à la création d'une connexion à la base de données.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

Par exemple, si **OSuser()** renvoie *WA\sbt*, la chaîne est évaluée comme `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, puis envoyée à la base de données lorsque la connexion est créée.

SQLQueryPrefix
Cette chaîne est envoyée pour chaque requête effectuée.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery Variables de caractère

DirectFieldColumnDelimiter

Vous pouvez définir le caractère utilisé comme délimiteur de champs dans les instructions **Direct Query** pour les bases de données nécessitant un autre caractère que la virgule. Le caractère spécifié doit être placé entre des guillemets simples dans l'instruction **SET**.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Vous pouvez spécifier le caractère à utiliser pour mettre des chaînes entre guillemets dans une requête générée. Le caractère par défaut est un guillemet simple. Le caractère spécifié doit être placé entre des guillemets simples dans l'instruction **SET**.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

Vous pouvez spécifier l'utilisation de la mise entre guillemets non-ANSI des identificateurs dans les requêtes générées. À l'heure actuelle, le seul type de mise entre guillemets non-ANSI disponible est GoogleBQ. La norme ANSI est définie par défaut. Il est possible d'employer des majuscules, des minuscules et une casse mixte (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Par exemple, la mise entre guillemets ANSI est utilisée dans l'instruction **SELECT** suivante :

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Lorsque la variable **DirectIdentifierQuoteStyle** est définie sur "GoogleBQ", l'instruction **SELECT** applique la mise entre guillemets suivante :

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Vous pouvez spécifier le caractère destiné à contrôler la mise entre guillemets des identificateurs dans une requête générée. Il peut s'agir soit d'un caractère (tel qu'un guillemet double) ou de deux (tels qu'une paire de crochets). Le caractère par défaut est un guillemet double.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold

Lorsque des champs Direct Discovery sont utilisés dans une visualisation de type **Table**, un seuil est défini en vue de limiter le nombre de lignes affichées. Le seuil par défaut est de 1 000 enregistrements. Il est possible de modifier le paramètre de seuil par défaut en définissant la variable **DirectTableBoxListThreshold** dans le script de chargement. Par exemple :

```
SET DirectTableBoxListThreshold=5000;
```

Le paramètre de seuil s'applique uniquement aux visualisations de type **Table** contenant des champs Direct Discovery. Les visualisations de type **Table** comportant exclusivement des champs chargés en mémoire ne sont pas limitées par le paramètre **DirectTableBoxListThreshold**.

Aucun champ n'est affiché dans la visualisation de type **Table** tant que la sélection comporte un nombre d'enregistrements supérieur à la limite de seuil fixée.

Variables d'interprétation des nombres Direct Discovery

DirectMoneyDecimalSep

Le séparateur décimal défini remplace le symbole décimal de la devise dans l'instruction SQL générée pour charger les données à l'aide de la fonction Direct Discovery. Ce caractère doit correspondre au caractère utilisé dans **DirectMoneyFormat**.

La valeur par défaut est `'.'`.

Exemple :

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

Le symbole défini remplace le format monétaire dans l'instruction SQL générée pour charger les données à l'aide de la fonction Direct Discovery. Le symbole de devise pour le séparateur de milliers doit être exclu.

La valeur par défaut est `'#.0000'`.

Exemple :

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

Le format de l'heure défini remplace le format de l'heure spécifié dans l'instruction SQL générée pour le chargement des données à l'aide de la fonction Direct Discovery.

Exemple :

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

DirectDateFormat

Le format de date défini remplace le format de date spécifié dans l'instruction SQL générée pour le chargement des données à l'aide de la fonction Direct Discovery.

Exemple :

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

DirectTimeStampFormat

Le format défini remplace le format de date et heure spécifié dans l'instruction SQL générée pour le chargement des données à l'aide de la fonction Direct Discovery.

Exemple :

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

2.16 Variables d'erreur

Les valeurs de toutes les variables d'erreur sont conservées après l'exécution du script. La première variable, ErrorMode, provient de l'utilisateur tandis que les trois dernières proviennent de Qlik Sense et présentent des informations sur les erreurs contenues dans le script.

Vue d'ensemble des variables d'erreur

Chaque variable est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la variable qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Consultez l'aide en ligne de Qlik Sense pour en savoir plus sur les variables.

ErrorMode

Cette variable d'erreur détermine l'action que Qlik Sense doit entreprendre s'il rencontre une erreur au cours de l'exécution du script.

```
ErrorMode
```

ScriptError

Cette variable d'erreur renvoie le code d'erreur de la dernière instruction de script exécutée.

```
ScriptError
```

ScriptErrorCount

Cette variable d'erreur renvoie le nombre total d'instructions ayant généré des erreurs au cours de l'exécution du script actif. Cette variable est toujours réinitialisée sur 0 au début de l'exécution du script.

```
ScriptErrorCount
```

ScriptErrorList

Cette variable d'erreur contient une liste concaténée de toutes les erreurs de script qui se sont produites au cours de la dernière exécution du script. Les erreurs sont séparées par des sauts de ligne.

```
ScriptErrorList
```

ErrorMode

Cette variable d'erreur détermine l'action que Qlik Sense doit entreprendre s'il rencontre une erreur au cours de l'exécution du script.

Syntaxe :

```
ErrorMode
```

Arguments :

Arguments

Argument	Description
ErrorMode=1	Paramètre par défaut. L'exécution du script est interrompue et l'utilisateur est invité à choisir une action (mode autre que le traitement par lot).
ErrorMode =0	Qlik Sense ignore simplement l'erreur et poursuit l'exécution du script à l'instruction suivante.
ErrorMode =2	Qlik Sense génère immédiatement un message d'erreur indiquant que l'exécution du script a échoué, sans inviter l'utilisateur à choisir une action au préalable.

Exemple :

```
set ErrorMode=0;
```

ScriptError

Cette variable d'erreur renvoie le code d'erreur de la dernière instruction de script exécutée.

Syntaxe :

```
ScriptError
```

Cette variable est réinitialisée sur 0 après chaque instruction de script exécutée correctement. Si une erreur se produit, elle est définie sur un code d'erreur Qlik Sense interne. Les codes d'erreur sont des valeurs doubles comportant un composant numérique et un composant textuel. Les codes d'erreur sont les suivants :

Codes d'erreur de script

Code d'erreur	Description
0	Pas d'erreur. Le texte de la valeur double est vide.
1	Erreur générale.
2	Erreur de syntaxe.
3	Erreur générale ODBC.
4	Erreur générale OLE DB.
5	Erreur générale de base de données personnalisée.
6	Erreur générale XML.
7	Erreur générale HTML.
8	Fichier introuvable.
9	Base de données introuvable.
10	Table introuvable.
11	Champ introuvable.
12	Format de fichier erroné.
16	Erreur sémantique.

Exemple :

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Cette variable d'erreur renvoie le nombre total d'instructions ayant généré des erreurs au cours de l'exécution du script actif. Cette variable est toujours réinitialisée sur 0 au début de l'exécution du script.

Syntaxe :

```
ScriptErrorCount
```

ScriptErrorList

Cette variable d'erreur contient une liste concaténée de toutes les erreurs de script qui se sont produites au cours de la dernière exécution du script. Les erreurs sont séparées par des sauts de ligne.

Syntaxe :

```
ScriptErrorList
```

2 Expressions de script

Vous pouvez utiliser des expressions dans les instructions **LOAD** comme dans les instructions **SELECT**. La syntaxe et les fonctions décrites dans cette section s'appliquent à l'instruction **LOAD** et pas à l'instruction **SELECT**, puisque cette dernière est interprétée par le pilote ODBC et pas par Qlik Sense. Cependant, la plupart des pilotes ODBC sont capables d'interpréter un grand nombre des fonctions décrites ci-dessous.

Les expressions se composent de fonctions, de champs et d'opérateurs, combinés dans une syntaxe.

Toutes les expressions d'un script Qlik Sense renvoient un nombre et/ou une chaîne, selon le cas. Les opérateurs et les fonctions logiques renvoient 0 pour False et -1 pour True. Les conversions de nombres en chaînes et inversement sont implicites. Les opérateurs et les fonctions logiques interprètent 0 comme False et toutes les autres valeurs comme True.

La syntaxe générale d'une expression est la suivante :

Syntaxe générale

Expression	Champs	Opérateur
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

où :

- **constant** est une chaîne (texte, date ou heure) placée entre guillemets simples ou un nombre. Les constantes sont écrites sans séparateur de milliers et avec un point comme séparateur décimal.
- **fieldref** correspond au nom d'un champ de la table chargée.
- **operator1** est un opérateur unaire (qui agit sur une seule expression, celle qui se trouve à droite).
- **operator2** est un opérateur binaire (qui agit sur deux expressions, une de chaque côté).
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Le nombre et les types de paramètres ne sont pas arbitraires. Ils dépendent de la fonction utilisée.

Les expressions et les fonctions peuvent ainsi être imbriquées librement ; tant que l'expression renvoie une valeur interprétable, Qlik Sense ne génère pas de messages d'erreur.

3 Expressions de graphique

Une expression (visualisation) de graphique est une combinaison de fonctions, de champs, d'opérateurs mathématiques (+ * / =) et d'autres mesures. Les expressions permettent de traiter les données contenues dans l'application afin de générer un résultat pouvant être affiché dans une visualisation. Leur utilisation ne se limite pas aux mesures. Vous pouvez créer des visualisations plus dynamiques et attrayantes en employant des expressions dans les titres, les sous-titres, les notes de bas de page et même les dimensions.

Autrement dit, au lieu d'utiliser, par exemple, un texte statique comme titre dans une visualisation, il est possible de définir le titre comme une expression dont le résultat change en fonction des sélections effectuées.

Pour obtenir des références détaillées sur les fonctions de script et de graphique, consultez le Syntaxe des scripts et fonctions de graphique.

3.1 Définition de l'étendue d'une agrégation

Il existe généralement deux facteurs qui, ensemble, déterminent les enregistrements utilisés pour définir la valeur d'une agrégation dans une expression. Lorsque vous travaillez dans les visualisations, ces facteurs sont les suivants :

- Valeur dimensionnelle (de l'agrégation dans une expression de graphique)
- Sélections

Ensemble, ces facteurs définissent l'étendue de l'agrégation. Dans certaines situations, il peut s'avérer souhaitable de ne pas prendre en compte la sélection, la dimension (ou les deux) dans le calcul. Dans les fonctions de graphique, pour ce faire, vous utilisez le qualificateur TOTAL, l'analyse d'ensembles ou une combinaison des deux.

Agrégation : Méthode et description

Méthode	Description
Qualificateur TOTAL	Si vous utilisez le qualificateur total à l'intérieur de votre fonction d'agrégation, la valeur dimensionnelle n'est pas prise en compte. L'agrégation est appliquée à toutes les valeurs de champ possibles. Le qualificateur **TOTAL** peut être suivi d'une liste d'un ou de plusieurs noms de champ placés entre crochets angulaires. Ces noms de champ doivent constituer un sous-ensemble des variables de dimension du graphique. Dans ce cas, toutes les variables de dimension du graphique sont ignorées lors du calcul, à l'exception de celles figurant dans la liste. Autrement dit, une valeur est renvoyée pour chaque combinaison de valeurs de champ dans les champs de dimension de la liste. Il est par ailleurs possible d'inclure dans la liste des champs qui ne constituent pas une dimension dans un graphique. Cette option peut s'avérer utile dans le cas de dimensions groupées, où les champs de dimension ne sont pas fixes. Si vous listez toutes les variables du groupe, la fonction se déclenche lors de tout changement de niveau hiérarchique.
Analyse d'ensembles	L'utilisation de l'analyse d'ensembles à l'intérieur de l'agrégation remplace la sélection. L'agrégation est appliquée à toutes les valeurs réparties sur les dimensions.
Qualificateur TOTAL et analyse d'ensembles	Si vous utilisez le qualificateur **TOTAL** et l'analyse d'ensembles à l'intérieur de l'agrégation, la sélection est remplacée et les dimensions ne sont pas prises en compte.

Qualificateur ALL	Si vous utilisez le qualificateur **ALL** à l'intérieur de l'agrégation, la sélection et les dimensions ne sont pas prises en compte. Il est possible de parvenir au même résultat avec l'instruction d'analyse d'ensembles {1} et le qualificateur **TOTAL** :
	
	`=sum(All Sales)`
	
	`=sum({1} Total Sales)`

Exemple : Qualificateur TOTAL

L'exemple suivant illustre la manière dont le qualificateur TOTAL peut servir à calculer une part relative. Supposons que Q2 ait été sélectionné. L'utilisation de TOTAL permet alors de calculer la somme de toutes les valeurs sans tenir compte des dimensions.

Exemple : Qualificateur Total

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100%
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

*Afin d'afficher les nombres sous forme de pourcentage, dans le panneau des propriétés, pour la mesure concernée, sous **Number formatting**, sélectionnez **Number**, puis dans le menu déroulant **Formatting**, choisissez **Simple** et l'un des formats de %.*

Exemple : Analyse d'ensembles

L'exemple suivant illustre l'utilisation de ce type d'analyse en vue d'établir une comparaison entre des ensemble de données avant toute sélection. Supposons que Q2 ait été sélectionné. L'utilisation de l'analyse d'ensembles avec la définition de l'ensemble {1} permet de calculer la somme de toutes les valeurs sans tenir compte des sélections mais en suivant la répartition d'après les dimensions.

Exemple : Analyse d'ensembles

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100%
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

Exemple : Qualificateur TOTAL et analyse d'ensembles

L'exemple suivant illustre l'utilisation combinée de l'analyse d'ensembles et du qualificateur TOTAL en vue d'établir une comparaison entre des ensembles de données portant sur la totalité des dimensions et avant toute sélection. Supposons que Q2 ait été sélectionné. L'utilisation de l'analyse d'ensembles avec la définition de l'ensemble {1} et le qualificateur TOTAL permet de calculer la somme de toutes les valeurs sans tenir compte des sélections et des dimensions.

Exemple : Qualificateur TOTAL et analyse d'ensembles

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%

Données utilisées dans les exemples :

```
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 Analyse d'ensembles

Lorsque vous effectuez une sélection dans une application, vous définissez un sous-ensemble d'enregistrements dans les données. Les fonctions d'agrégation telles que `Sum()`, `Max()`, `Min()`, `Avg()` et `Count()` sont calculées en fonction de ce sous-ensemble.

En d'autres termes, votre sélection définit l'étendue de l'agrégation ; elle définit l'ensemble des enregistrements sur lesquels les calculs sont effectués.

L'analyse d'ensembles est une manière de définir une étendue différente de l'ensemble d'enregistrements défini par la sélection active. Cette nouvelle étendue peut également être considérée comme une sélection alternative.

Cela peut s'avérer utile si vous souhaitez comparer la sélection active à une valeur donnée, par exemple, à la valeur de l'an dernier ou à la part de marché mondiale.

Expressions d'ensemble

Les expressions d'ensemble peuvent être utilisées à l'intérieur et à l'extérieur des fonctions d'agrégation et sont encadrées par des accolades.

Exemple : Expression d'ensemble interne

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

Exemple : Expression d'ensemble externe

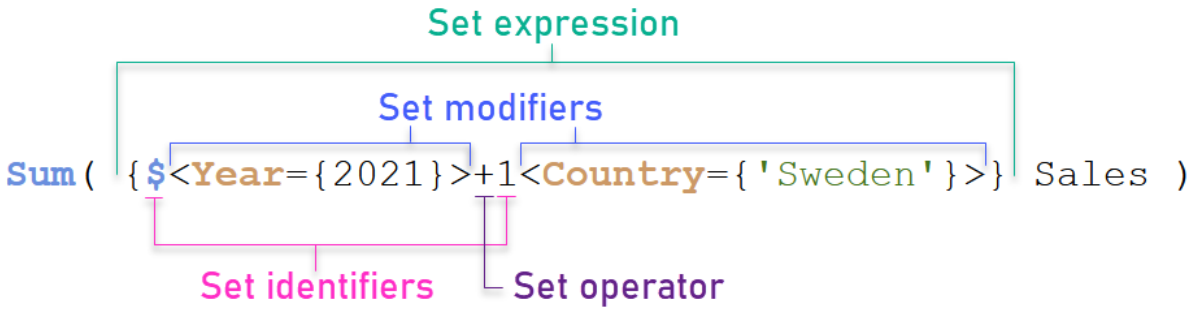
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Une expression d'ensemble se compose d'une combinaison des éléments suivants :

- **Identifiers**. Un identificateur d'ensemble représente une sélection, définie ailleurs. Il représente également un ensemble spécifique d'enregistrements dans les données. Il peut s'agir de la sélection active, d'une sélection d'un favori ou d'une sélection d'un état alternatif. Une expression d'ensemble simple se compose d'un seul identificateur, par exemple le signe dollar, {$}, qui signifie tous les enregistrements de la sélection active.
  Exemples : `$`, `1`, `BookMark1`, `State2`
- **Operators**. Un opérateur d'ensemble peut être utilisé pour créer des unions, des différences ou des intersections entre différents identificateurs d'ensemble. Vous pouvez ainsi créer un sous-ensemble ou un super-ensemble des sélections définies par les identificateurs d'ensemble.
  Exemples : `+`, `-`, `*`, `/`
- **Modifiers**. Un modificateur d'ensemble peut être ajouté à l'identificateur d'ensemble pour modifier sa sélection. Un modificateur peut également être utilisé indépendamment ; il modifiera alors l'identificateur par défaut. Un modificateur doit être encadré par des crochets angulaires `<…>`.
  Exemples : `<Year={2020}>`, `<Supplier={ACME}>`

Les éléments sont combinés pour former des expressions d'ensemble.

Éléments d'une expression d'ensemble



L'expression d'ensemble ci-dessus, par exemple, est créée à partir de l'agrégation `Sum(Sales)`.

Le premier opérande renvoie sales pour l'année `2021` pour la sélection active, tel qu'indiqué par l'identificateur d'ensemble `$` et le modificateur d'ensemble contenant la sélection de l'année `2021`. Le deuxième opérande renvoie `Sales` pour `Sweden` et ignore la sélection active, tel qu'indiqué par l'identificateur d'ensemble`1`.

Pour finir, l'expression renvoie un ensemble composé des enregistrements qui appartiennent à l'un des deux opérandes d'ensemble, tel qu'indiqué par l'opérateur d'ensemble `+`.

Exemples

Des exemples combinant des éléments d'expression d'ensemble ci-dessus sont disponibles dans les rubriques suivantes :

Ensembles naturels

En règle générale, une expression d'ensemble représente un ensemble d'enregistrements du modèle de données et une sélection qui définit ce sous-ensemble de données. Dans ce cas, l'ensemble est appelé ensemble naturel.

Les identificateurs d'ensemble, avec ou sans modificateurs d'ensemble, représentent toujours des ensembles naturels.

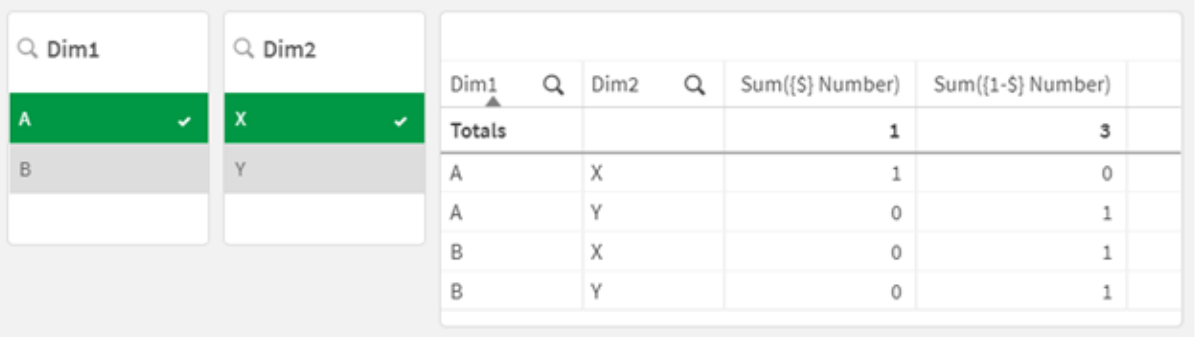
Cependant, une expression d'ensemble qui utilise des opérateurs d'ensemble représente également un sous-ensemble des enregistrements, mais ne peut généralement tout de même pas être décrite via une sélection de valeurs de champ. Une telle expression est un ensemble non naturel.

Par exemple, l'ensemble fourni par `{1-$}` ne peut pas toujours être défini par une sélection. Il ne s'agit donc pas d'un ensemble naturel. Cela peut être mis en évidence en chargeant les données suivantes, en les ajoutant à une table, puis en effectuant des sélectons via des volets de filtre.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Si vous effectuez des sélections pour `Dim1` et pour `Dim2`, vous obtenez la vue indiquée dans le tableau suivant.

Tableau avec des ensembles naturels et non naturels



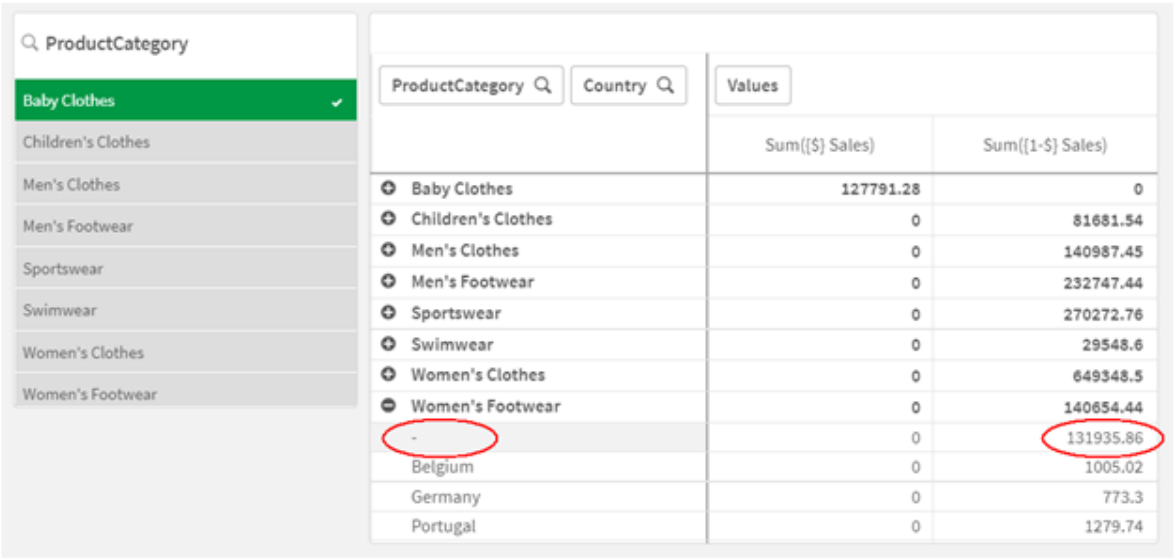
L'expression d'ensemble de la première mesure utilise un ensemble naturel : elle correspond à la sélection effectuée par `{$}`.

La deuxième mesure est différente. Elle utilise `{1-$}`. Il n'est pas possible d'effectuer une sélection correspondant à cet ensemble ; il s'agit donc d'un ensemble non naturel.

Cette distinction entraîne un certain nombre de conséquences :

- Les modificateurs d'ensemble peuvent être appliqués uniquement à des identificateurs d'ensemble. Ils ne peuvent pas être appliqués à une expression d'ensemble arbitraire. Par exemple, il n'est pas possible d'utiliser une expression d'ensemble telle que :
  `{ (BM01 * BM02) <Field={x,y}> }`
  Ici, les parenthèses (rondes) normales impliquent l'évaluation de l'intersection entre `BM01` et `BM02` avant l'application du modificateur d'ensemble. Cela est dû au fait qu'il n'existe aucun ensemble d'éléments susceptible d'être modifié.
- Vous ne pouvez pas utiliser des ensembles non naturels à l'intérieur des fonctions d'élément `P()` et `E()`. Ces fonctions renvoient un ensemble d'éléments, mais il n'est pas possible de déduire l'ensemble d'éléments d'un ensemble non naturel.
- Une mesure utilisant un ensemble non naturel ne peut pas toujours être attribuée à la valeur dimensionnelle appropriée si le modèle de données contient de nombreuses tables. Par exemple, dans le graphique suivant, certains chiffres des ventes exclus sont attribués à la valeur Country correcte, tandis que d'autres ont la valeur NULL pour Country.

Graphique avec un ensemble non naturel



Le fait que l'affectation soit effectuée correctement ou non dépend du modèle de données. Dans ce cas, le chiffre ne peut pas être affecté s'il correspond à un pays exclu de la sélection.

Identificateur	Description
1	Représente l'ensemble complet de tous les enregistrements dans l'application, indépendamment des sélections effectuées.
$	Représente les enregistrements de la sélection active. L'expression d'ensemble **{$}** revient donc à ne pas définir d'expression.
$1	Représente la sélection précédente. $2 représente l'avant-dernière sélection, et ainsi de suite.
$_1	Représente la sélection suivante (prochaine). $_2 représente la sélection suivante sauf une, et ainsi de suite.
BM01	Vous pouvez utiliser n'importe quel ID ou nom de favori.
MyAltState	Vous pouvez référencer les sélections effectuées dans un état alternatif par le nom de cet état.

Exemple	Résultat
sum ({1} Sales)	Renvoie le total des ventes pour l'application en ignorant les sélections mais pas la dimension.
sum ({$} Sales)	Renvoie les ventes pour la sélection active, ce qui équivaut à sum(Sales).
sum ({$1} Sales)	Renvoie les ventes pour la sélection précédente.
sum ({BM01} Sales)	Renvoie les ventes pour le favori intitulé *BM01*

Exemple	Résultat
sum({$<OrderDate = DeliveryDate>} Sales)	Renvoie les ventes pour la sélection active, où OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Renvoie les ventes pour la région USA sans tenir compte de la sélection active.
sum({$<Region = >} Sales)	Renvoie les ventes pour la sélection en ayant supprimé la sélection effectuée dans '*Region*'.
sum({<Region = >} Sales)	Renvoie les mêmes résultats que l'exemple ci-dessus. Lorsque l'ensemble à modifier est omis, la fonction utilise $.
sum({$<Year={2000}, Region={"U*"}>} Sales)	Renvoie les ventes pour la sélection active, mais en utilisant de nouvelles sélections dans les champs *Year* et *Region*.

Identificateurs d'ensemble

Un identificateur d'ensemble représente un ensemble d'enregistrements dans les données ; soit toutes les données, soit un sous-ensemble d'entre elles. Il s'agit de l'ensemble d'enregistrements défini par une sélection. Il peut s'agir de la sélection active, de toutes les données (pas de sélection), d'une sélection d'un favori ou d'une sélection d'un état alternatif.

Dans l'exemple `Sum( {$<Year = {2009}>} Sales )`, l'identificateur est le signe dollar : `$`. Cela représente la sélection active. Cela représente également tous les enregistrements possibles. Cet ensemble peut ensuite être altéré par la partie modificateur de l'expression d'ensemble : la sélection `2009` de `Year` est ajoutée.

Dans une expression d'ensemble plus complexe, il est possible d'utiliser ensemble deux identificateurs avec un opérateur pour former une union, une différence ou une intersection des deux ensembles d'enregistrements.

Le tableau suivant illustre quelques identificateurs courants.

Exemples présentant des identificateurs courants

Identificateur	Description
`1`	Représente l'ensemble complet de tous les enregistrements dans l'application, indépendamment des sélections effectuées.
`$`	Représente les enregistrements de la sélection active dans l'état par défaut. L'expression d'ensemble `{$}` revient donc généralement à ne pas définir d'expression d'ensemble.
`$1`	Représente la sélection précédente dans l'état par défaut. `$2` représente la sélection précédant la sélection précédente, etc.
`$_1`	Représente la sélection suivante. `$_2` représente la sélection suivant la sélection suivante, etc.

Identificateur	Description
`BM01`	Vous pouvez utiliser n'importe quel ID ou nom de favori.
`AltState`	Vous pouvez référencer un état alternatif par le nom de son état.
`AltState::BM01`	Un favori contient les sélections de tous les états, et vous pouvez référencer un favori spécifique en qualifiant le nom du favori.

Le tableau suivant illustre des exemples présentant différents identificateurs.

Exemples présentant différents identificateurs

Exemple	Résultat
`Sum ({1} Sales)`	Renvoie le total des ventes pour l'application en ignorant les sélections mais pas la dimension.
`Sum ({$} Sales)`	Renvoie les ventes pour la sélection active, ce qui équivaut à `Sum(Sales)`.
`Sum ({$1} Sales)`	Renvoie les ventes pour la sélection précédente.
`Sum ({BM01} Sales)`	Renvoie les ventes pour le favori intitulé `BM01`

Opérateurs d'ensemble

Les opérateurs d'ensemble permettent d'inclure, d'exclure ou d'intersecter des ensembles de données. Tous les opérateurs utilisent les ensembles comme opérandes et renvoient un ensemble pour résultat.

Vous pouvez utiliser des opérateurs d'ensemble dans deux situations différentes :

- Pour effectuer une opération d'ensemble sur des identificateurs d'ensemble représentant des ensembles d'enregistrements dans des données.
- Pour effectuer une opération d'ensemble sur les ensembles d'éléments, sur les valeurs de champ ou à l'intérieur d'un modificateur d'ensemble.

Le tableau suivant illustre les opérateurs qui peuvent être utilisés dans des expressions d'ensemble.

Opérateurs

Opérateur	Description
`+`	Union. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant à l'un ou l'autre des deux opérandes d'ensemble.
`-`	Exclusion. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant au premier opérande d'ensemble, mais pas à l'autre opérande d'ensemble des deux opérandes d'ensemble. Lorsqu'il est utilisé en tant qu'opérateur unaire, il renvoie un ensemble complémentaire.
`*`	Intersection. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant aux deux opérandes d'ensemble.

Opérateur	Description
/	Différence symétrique (XOR). Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant à l'un ou à l'autre des deux opérandes d'ensemble, mais pas aux deux.

Le tableau suivant illustre des exemples comprenant des opérateurs.

Exemples comportant des opérateurs

Exemple	Résultat
`Sum ({1-$} Sales)`	Renvoie les ventes pour tous les éléments exclus par la sélection active.
`Sum ({$*BM01} Sales)`	Renvoie les ventes pour l'intersection entre la sélection et le favori `BM01`.
`Sum ({-($+BM01)} Sales)`	Renvoie les ventes exclues par la sélection et le favori `BM01`.
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Renvoie les ventes de l'année 2009 associées aux sélections actives et ajoute l'ensemble de données complet associé au pays `Sweden` pour toutes les années.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Renvoie les ventes (sales) des pays qui commencent par un `S` ou qui se terminent par `land`.

Modificateurs d'ensemble

Les expressions d'ensemble sont utilisées pour définir l'étendue d'un calcul. La partie centrale de l'expression d'ensemble est le modificateur d'ensemble qui spécifie une sélection. Cela est utilisé pour modifier la sélection de l'utilisateur, ou la sélection de l'identificateur d'ensemble, et le résultat définit une nouvelle étendue pour le calcul.

Le modificateur d'ensemble se compose d'un ou de plusieurs noms de champ, chacun suivi d'une sélection devant être effectuée dans le champ. Le modificateur est encadré par des crochets angulaires : < >

Par exemple :

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Ensembles d'éléments

Un ensemble d'éléments peut être défini via les éléments suivants :

- Une liste de valeurs
- Une recherche
- Une référence à un autre champ
- Une fonction d'ensemble

Si la définition de l'ensemble d'éléments est omise, le modificateur d'ensemble efface toute sélection de ce champ. Par exemple :

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

Exemples : Expressions de graphique pour modificateurs d'ensemble basés sur des ensembles d'éléments

Exemples - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
MyTable:
Load * Inline [
Country, Year, Sales
Argentina, 2014, 66295.03
Argentina, 2015, 140037.89
Austria, 2014, 54166.09
Austria, 2015, 182739.87
Belgium, 2014, 182766.87
Belgium, 2015, 178042.33
Brazil, 2014, 174492.67
Brazil, 2015, 2104.22
Canada, 2014, 101801.33
Canada, 2015, 40288.25
Denmark, 2014, 45273.25
Denmark, 2015, 106938.41
Finland, 2014, 107565.55
Finland, 2015, 30583.44
France, 2014, 115644.26
France, 2015, 30696.98
Germany, 2014, 8775.18
Germany, 2015, 77185.68
];
```

Expressions de graphique

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

Tableau - modificateurs d'ensemble basés sur des ensembles d'éléments

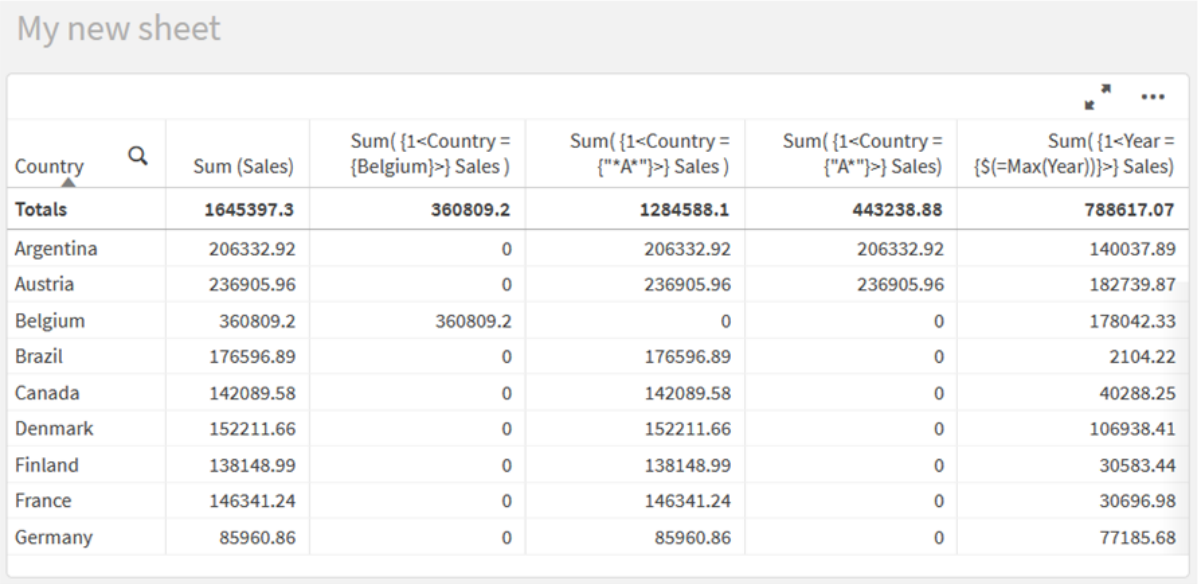
Country	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum ({1<Year= {$(=Max (Year))}>} Sales)
Totals	1645397.3	360809.2	1284588.1	443238.88	788617.07

Country	Sum(Sales)	Sum ({1<Country={Belgium}>} Sales)	Sum ({1<Country={"*A*"}>} Sales)	Sum ({1<Country={"A*"}>} Sales)	Sum ({1<Year={$(=Max(Year))}>} Sales)
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Sales)`
    somme `Sales` sans expression d'ensemble.
  - `Sum({1<Country={Belgium}>}Sales)`
    Sélectionne `Belgium`, puis la somme correspondante `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Sélectionne tous les pays qui contiennent un `A`, puis la somme correspondante `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Sélectionne tous les pays qui commencent par un `A`, puis la somme correspondante `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Calculez `Max(Year)`, qui est `2015`, puis la somme correspondante `Sales`.

Modificateurs d'ensemble basés sur des ensembles d'éléments


My new sheet

Country	Sum (Sales)	Sum({1<Country = {Belgium}>} Sales)	Sum({1<Country = {"*A*"}>} Sales)	Sum({1<Country = {"A*"}>} Sales)	Sum({1<Year = {$(=Max(Year))}>} Sales)
Totals	**1645397.3**	**360809.2**	**1284588.1**	**443238.88**	**788617.07**
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Valeurs répertoriées

L'exemple d'ensemble d'éléments le plus courant est celui basé sur une liste de valeurs de champ entre accolades. Par exemple :

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

Les accolades internes définissent l'ensemble d'éléments. Les valeurs individuelles sont séparées par des virgules.

Guillemets et sensibilité à la casse

Si les valeurs contiennent des espaces ou des caractères spéciaux, elles doivent être encadrées par des guillemets. Les guillemets simples seront une correspondance littérale sensible à la casse à une seule valeur de champ. Les guillemets doubles impliquent une correspondance non sensible à la casse à une ou plusieurs valeurs de champ. Par exemple :

- `<Country = {'New Zealand'}>`
  Correspond uniquement à `New Zealand`.
- `<Country = {"New Zealand"}>`
  Correspond à `New Zealand`, à `NEW ZEALAND` et à `new zealand`.

Les dates doivent être encadrées par des guillemets et utiliser le format de date du champ en question. Par exemple :

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Les guillemets doubles peuvent être remplacés par des crochets ou par des accents graves.

Recherches

Il est également possible de créer des ensembles d'éléments via des recherches. Par exemple :

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Les caractères génériques peuvent être utilisés dans une recherche de texte : Un astérisque (`*`) représente n'importe quel nombre de caractères et un point d'interrogation (`?`) représente un seul caractère. Les opérateurs relationnels peuvent être utilisés pour définir des recherches numériques.

Pour les recherches, vous devez toujours utiliser des guillemets doubles. Les recherches ne sont pas sensibles à la casse des caractères.

Expansions dollar

Les expansions dollar sont nécessaires si vous souhaitez utiliser un calcul à l'intérieur de votre ensemble d'éléments. Par exemple, si vous souhaitez examiner uniquement la dernière année possible, vous pouvez utiliser :

```
<Year = {$(=Max(Year))}>
```

Valeurs sélectionnées dans d'autres champs

Les modificateurs peuvent être basés sur les valeurs sélectionnées d'un autre champ. Par exemple :

```
<OrderDate = DeliveryDate>
```

Ce modificateur récupère les valeurs sélectionnées à partir de `DeliveryDate` pour les appliquer en tant que sélection à `OrderDate`. Si vous disposez de nombreuses valeurs distinctes (plus de deux cents), cette opération est déconseillée, car elle mobilise énormément les ressources du processeur.

Fonctions d'un ensemble d'éléments

Un ensemble d'éléments peut également être basé sur les fonctions d'ensemble `P()` (valeurs possibles) et `E()` (valeurs exclues).

Par exemple, si vous souhaitez sélectionner les pays dans lesquels le produit `Cap` a été vendu, vous pouvez utiliser :

```
<Country = P({1<Product={Cap}>} Country)>
```

De même, si vous souhaitez sélectionner les pays dans lesquels le produit `Cap` n'a pas été vendu, vous pouvez utiliser :

```
<Country = E({1<Product={Cap}>} Country)>
```

Modificateurs d'ensemble associés à des recherches

Vous pouvez créer des ensembles d'éléments via des recherches avec des modificateurs d'ensemble.

Par exemple :

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Les recherches doivent toujours être encadrées par des guillemets doubles, des crochets ou des accents graves. Vous pouvez utiliser une liste avec un mélange de chaînes littérales (guillemets simples) et de recherches (guillemets doubles). Par exemple :

```
<Product = {'Nut', "*Bolt", Washer}>
```

Recherches de texte

Les caractères génériques et d'autres symboles peuvent être utilisés dans les recherches de texte :

- Un astérisque (*) représentera n'importe quel nombre de caractères.
- Un point d'interrogation (?) représentera un seul caractère.
- Un accent circonflexe (^) marquera le début d'un mot.

Par exemple :

- `<Country = {"C*", "*land"}>`
  Correspond à tous les pays commençant par un `C` ou se terminant par `land`.
- `<Country = {"*^z*"}>`
  Cela correspondra à tous les pays contenant un terme qui commence par un `z`, par exemple, `New Zealand`.

Recherches numériques

Vous pouvez effectuer des recherches numériques via les opérateurs relationnels suivants : `>`, `>=`, `<` et `<=`.

Une recherche numérique commence toujours par un de ces opérateurs. Par exemple :

- `<Year = {">2015"}>`
  Correspond à 2016 et aux années suivantes.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Correspond à toutes les dates de 2015. Notez la syntaxe pour décrire une plage temporelle entre deux dates. Le format de date doit correspondre au format de date du champ en question.

Recherches par expression

Pour effectuer des recherches plus poussées, vous pouvez utiliser des recherches par expression. Une agrégation est alors évaluée pour chaque valeur de champ du champ de recherche. Toutes les valeurs pour lesquelles l'expression recherchée renvoie true sont sélectionnées.

Une recherche par expression commence toujours par un signe égal : `=`

Par exemple :

```
<Customer = {"=Sum(Sales)>1000"}>
```

Cela renverra tous les clients avec une valeur sales supérieure à 1 000. `Sum(Sales)` est calculé en fonction de la sélection active. Cela signifie que si vous avez une sélection dans un autre champ, par exemple, dans le champ `Product`, vous obtiendrez les clients qui ont rempli la condition sales pour les produits sélectionnés uniquement.

Si vous souhaitez que la condition soit indépendante de la sélection, vous devez utiliser set analysis (analyse d'ensembles) à l'intérieur de la chaîne de recherche. Par exemple :

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Les expressions après le signe égal seront interprétées comme une valeur booléenne. Cela signifie que si elles sont évaluées sur autre chose, n'importe quel valeur différente de zéro sera interprétée comme true, tandis que zéro et les chaînes seront interprétés comme false.

Guillemets

Lorsque les chaînes de recherche contiennent des espaces ou des caractères spéciaux, utilisez des guillemets. Les guillemets simples impliquent une correspondance littérale sensible à la casse à une seule valeur de champ. Les guillemets doubles impliquent une recherche non sensible à la casse correspondant potentiellement à plusieurs valeurs de champ.

Par exemple :

- `<Country = {'New Zealand'}>`
  Correspond uniquement à `New Zealand`.
- `<Country = {"New Zealand"}>`
  Correspond à `New Zealand`, à `NEW ZEALAND` et à `new zealand`.

Les guillemets doubles peuvent être remplacés par des crochets ou par des accents graves.

Dans les versions précédentes de Qlik Sense, il n'existait pas de distinction entre les guillemets simples et les guillemets doubles, et toutes les chaînes entre guillemets étaient traitées comme des recherches. Afin de garantir la compatibilité avec les anciennes versions, les applications créées dans des versions antérieures de Qlik Sense continueront à fonctionner de la même manière qu'auparavant. Les applications créées avec Qlik Sense November 2017 ou une version ultérieure respecteront la différence entre les deux types de guillemets.

Exemples : Expressions de graphique pour modificateurs d'ensemble associés à des recherches

Exemples - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Exemple 1 : Expressions de graphique associées à des recherches de texte

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

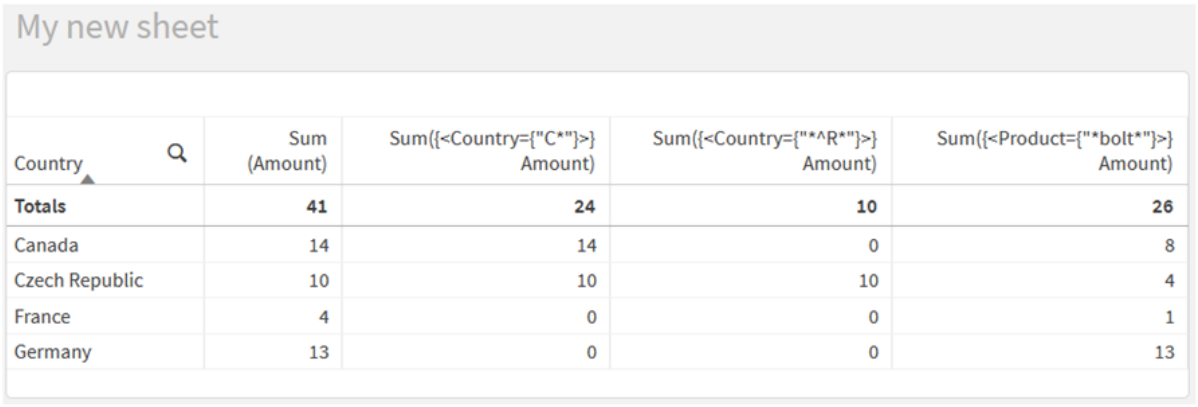
Tableau - modificateurs d'ensemble associés à des recherches de texte

Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<Country={"C*"}>}Amount)`
    Somme `Amount` de tous les pays qui commencent par un `C` tels que `Canada` et `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Somme `Amount` de tous les pays contenant un terme qui commence par un `R` tels que `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Somme `Amount` de tous les produits qui contiennent la chaîne `bolt` tels que `Bolt` et `Anchor bolt`.

Modificateurs d'ensemble associés à des recherches de texte



Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Exemple 2 : Expressions de graphique associées à des recherches numériques

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

Tableau - modificateurs d'ensemble associés à des recherches numériques

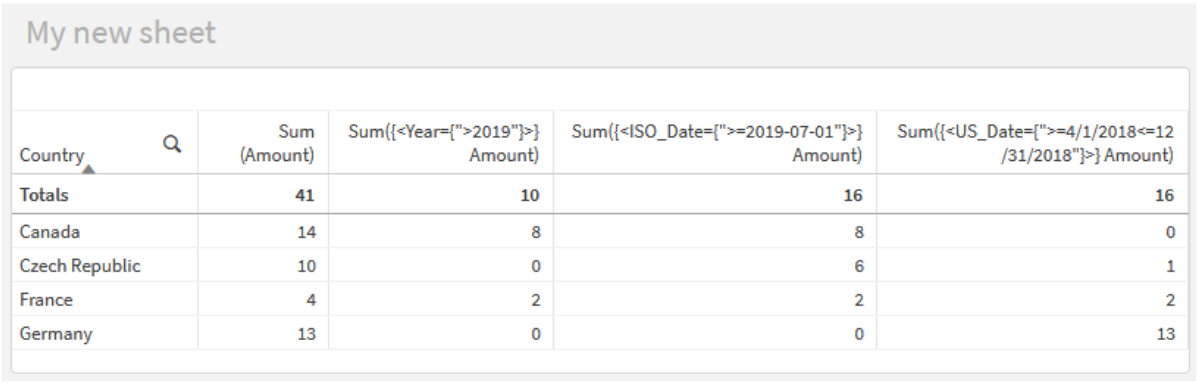
Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totals	**41**	**10**	**16**	**16**
Canada	14	8	8	0
Czech Republic	10	0	6	1

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

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<Year={">2019"}>}Amount)`
    Somme `Amount` de toutes les années après `2019`.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Somme `Amount` de toutes les dates le ou après le `2019-07-01`. Le format de la date de la recherche doit correspondre au format du champ.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Somme `Amount` de toutes les dates du `4/1/2018` au `12/31/2018`, y compris les dates de début et de fin. Le format des dates de la recherche doit correspondre au format du champ.

Modificateurs d'ensemble associés à des recherches numériques



Exemple 3 : Expressions de graphique associées à des recherches par expression

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

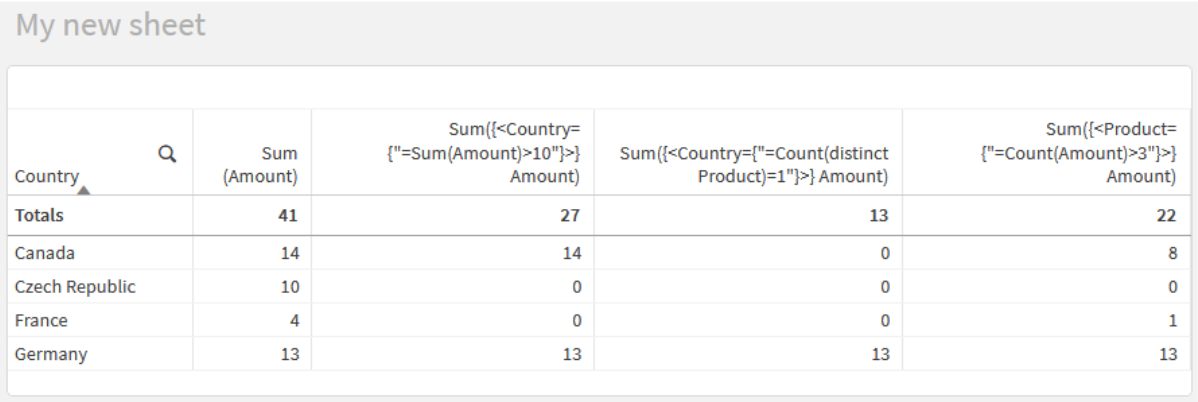
Table - Set modifiers with expression searches

Country	Sum (Amount)	Sum({<Country={"=Sum (Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count (Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Somme `Amount` de tous les pays qui ont une somme agrégée de `Amount` supérieure à `10`.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Somme `Amount` de tous les pays associés à exactement un produit distinct.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Somme `Amount` de tous les pays qui ont plus de trois transactions dans les données.

Modificateurs d'ensemble associés à des recherches par expression



Exemples	Résultats
sum({$-1<Product = {"*Internal*", "*Domestic*"}>} Sales)	Renvoie les ventes pour la sélection active, à l'exclusion des transactions relatives aux produits dont le nom contient la chaîne « Internal » ou « Domestic ».
sum({$<Customer = {"=Sum({1<Year = {2007}>} Sales) > 1000000"}>} Sales)	Renvoie les ventes pour la sélection active, mais avec une nouvelle sélection dans le champ Customer : seuls les clients dont le total des ventes était supérieur à 1 000 000 pour l'année 2007 sont retenus.

Modificateurs d'ensemble associés à des expansions de $

Les expansions de signe dollar sont des constructions qui sont calculées avant l'analyse et l'évaluation de l'expression. Le résultat est ensuite injecté dans l'expression au lieu de `$(…)`. Le calcul de l'expression est alors effectué via le résultat de l'expansion dollar.

L'éditeur d'expression affiche un aperçu de l'expansion dollar afin de vous permettre de vérifier par rapport à quoi votre expansion de signe dollar est évaluée.

Aperçu de l'expansion de signe dollar dans l'éditeur d'expression

Utilisez des expansions de signe dollar si vous souhaitez utiliser un calcul à l'intérieur de votre ensemble d'éléments.

Par exemple, si vous souhaitez examiner uniquement la dernière année possible, vous pouvez utiliser la construction suivante :

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` est calculé en premier, et le résultat est injecté dans l'expression au lieu de `$(…)`.

Le résultat après l'expansion dollar sera une expression telle que la suivante :

```
<Year = {2021}>
```

L'expression à l'intérieur de l'expansion dollar est calculée en fonction de la sélection active. Cela signifie que si vous avez une sélection dans un autre champ, le résultat de l'expression en sera affecté.

Si vous souhaitez que le calcul soit indépendant de la sélection, utilisez set analysis (analyse d'ensembles) à l'intérieur de l'expansion dollar. Par exemple :

```
<Year = {$(=Max({1} Year))}>
```

Chaînes

Pour que l'expansion dollar aboutisse à une chaîne, vous devez appliquer les règles normales en matière de guillemets. Par exemple :

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Le résultat après l'expansion dollar sera une expression telle que la suivante :

```
<Country = {'New Zealand'}>
```

Si vous n'utilisez pas les guillemets, vous obtiendrez une erreur de syntaxe.

Nombres

Si vous souhaitez que l'expansion dollar aboutisse à un nombre, assurez-vous que l'expansion a le même format que le champ. Cela signifie que vous devez parfois encadrer l'expression par une fonction de formatage.

Par exemple :

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Le résultat après l'expansion dollar sera une expression telle que la suivante :

```
<Amount = {12362.00}>
```

Utilisez un dièse pour forcer l'expansion à toujours utiliser un point décimal et aucun séparateur de milliers. Par exemple :

```
<Amount = {$(#=Max(Amount))}>
```

Dates

Si vous souhaitez que l'expansion dollar aboutisse à une date, assurez-vous que le format de l'expansion est correct. Cela signifie que vous devez parfois encadrer l'expression par une fonction de formatage.

Par exemple :

```
<Date = {'$(=Date(Max(Date)))'}>
```

Le résultat après l'expansion dollar sera une expression telle que la suivante :

```
<Date = {'12/31/2015'}>
```

Tout comme avec les chaînes, vous devez utiliser les guillemets corrects.

Un cas d'utilisation courant consiste à vouloir limiter le calcul au dernier mois (ou à la dernière année). Dans ce cas, vous pouvez utiliser une recherche numérique en combinaison avec la fonction `AddMonths()`.

Par exemple :

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Le résultat après l'expansion dollar sera une expression telle que la suivante :

```
<Date = {">=9/31/2021"}>
```

Cela sélectionnera tous les événements qui se sont produits le dernier mois.

Exemple : Expressions de graphique pour modificateurs d'ensemble associés à des expansions de signe dollar

Exemple - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
Let vToday = Today();
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2021-10-15, France, Washer, 1];
```

Expressions de graphique associées à des expansions de signe dollar

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

Tableau - modificateurs d'ensemble associés à des expansions $

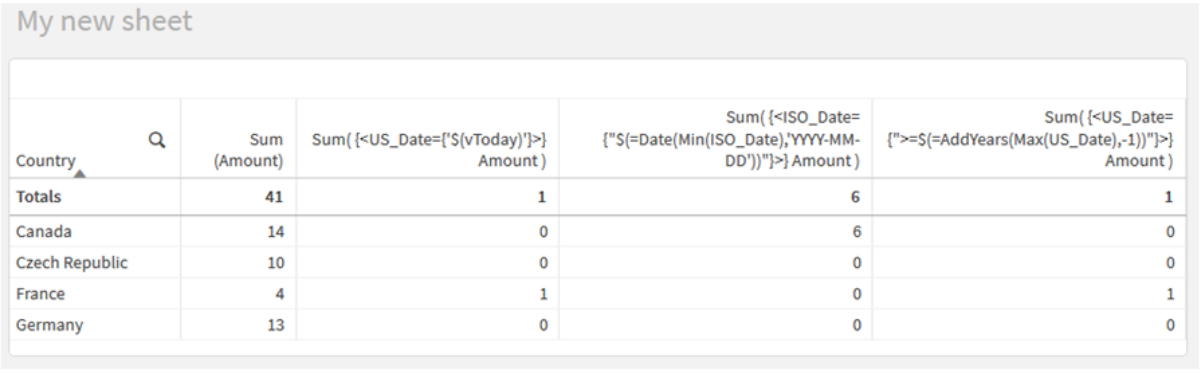
Country	Sum (Amount)	Sum({<US_ Date= {'$(vToday)'}>} Amount)	Sum({<ISO_Date= {"$(=Date(Min(ISO_ Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date= {">=$(=AddYears(Max (US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0

Country	Sum (Amount)	Sum({<US_ Date= {'$(vToday)'}>} Amount)	Sum({<ISO_Date= {"$(=Date(Min(ISO_ Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date= {">=$(=AddYears(Max (US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
France	4	1	0	1
Germany	13	0	0	0

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Somme `Amount` de tous les enregistrements dans lesquels la date `US_Date` est la même que dans la variable `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Somme `Amount` de tous les enregistrements dans lesquels la date `ISO_Date` est la même que la première valeur `ISO_Date` possible (la plus antérieure). La fonction `Date()` est nécessaire pour garantir que le format de la date correspond à celui du champ.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Somme `Amount` de tous les enregistrements qui contiennent une valeur `US_Date` après ou à la date une année avant la dernière date `US_Date` possible (la plus récente). La fonction `AddYears()` renverra une date au format spécifié par la variable `DateFormat`, et ce format doit correspondre au format du champ `US_Date`.

Modificateurs d'ensemble associés à des expansions de $



Exemples	Résultats
sum({$<Year = {$(#vLastYear)}>} Sales)	Renvoie les ventes pour l'année précédente par rapport à la sélection active. Dans cet exemple, une variable vLastYear contenant l'année en question est utilisée dans une expansion de $.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Renvoie les ventes pour l'année précédente par rapport à la sélection active. Dans cet exemple, l'expansion de dollar sert à calculer l'année précédente.

Modificateurs d'ensemble associés à des opérateurs d'ensemble

Les opérateurs d'ensemble permettent d'inclure, d'exclure ou d'intersecter différents ensembles d'éléments. Ils combinent les différentes méthodes pour définir des ensembles d'éléments.

Les opérateurs sont les mêmes que ceux utilisés pour les identificateurs d'ensemble.

Opérateurs

Opérateur	Description
+	Union. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant à l'un ou l'autre des deux opérandes d'ensemble.
-	Exclusion. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant au premier opérande d'ensemble, mais pas à l'autre opérande d'ensemble des deux opérandes d'ensemble. Lorsqu'il est utilisé en tant qu'opérateur unaire, il renvoie un ensemble complémentaire.
*	Intersection. Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant aux deux opérandes d'ensemble.
/	Différence symétrique (XOR). Cette opération binaire renvoie un ensemble composé des enregistrements ou des éléments appartenant à l'un ou à l'autre des deux opérandes d'ensemble, mais pas aux deux.

Par exemple, les deux modificateurs suivants définissent le même ensemble de valeurs de champ :

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Les deux expressions sélectionnent `1997` et les années qui commencent par `20`. En d'autres termes, il s'agit de l'union des deux conditions.

Les opérateurs d'ensemble permettent également des définitions plus complexes. Par exemple :

```
<Year = {1997, "20*"} - {2000}>
```

Cette expression sélectionnera les mêmes années que celles ci-dessus, mais, en plus, elle exclura l'année `2000`.

.

Exemples : Expressions de graphique pour modificateurs d'ensemble associés à des opérateurs d'ensemble

Exemples - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Expressions de graphique

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

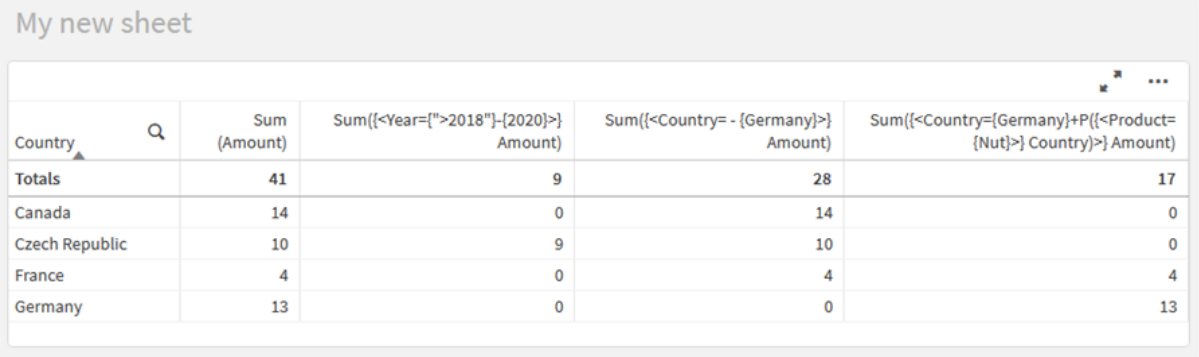
Tableau - modificateurs d'ensemble associés à des opérateurs d'ensemble

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Totals	**41**	**9**	**28**	**17**
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Somme `Amount` de toutes les années après `2018`, sauf `2020`.
  - `Sum({<Country=-{Germany}>}Amount)`
    Somme `Amount` de tous les pays sauf `Germany`. Notez l'opérateur d'exclusion unaire.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Somme `Amount` du pays `Germany` et de tous les pays associés au produit `Nut`.

Modificateurs d'ensemble associés à des opérateurs d'ensemble



Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	41	9	28	17
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Exemples	Résultats
sum({$<Product = Product + {OurProduct1} - {OurProduct2} >} Sales)	Renvoie les ventes pour la sélection active, mais avec le produit OurProduct1 ajouté à la liste des produits sélectionnés et OurProduct2 supprimé de la liste des produits sélectionnés.
sum({$<Year = Year + ({"20*",1997} - {2000}) >} Sales)	Renvoie les ventes pour la sélection active en utilisant des sélections supplémentaires dans le champ Year : 1997 et toutes celles qui commencent par 20 (sauf 2000). Vous noterez que si l'année 2000 fait partie de la sélection active, elle reste incluse après la modification.
sum({$<Year = (Year + {"20*",1997}) - {2000} >} Sales)	Renvoie presque les mêmes résultats que ci-dessus, mais 2000 est ici exclue, même si elle est initialement incluse dans la sélection active. L'exemple montre l'importance de l'utilisation des parenthèses pour définir un ordre de priorité.

Exemples	Résultats
sum({$<Year = {“*”} - {2000}, Product = {“*bearing*”} >} Sales)	Renvoie les ventes pour la sélection active en utilisant une nouvelle sélection dans le champ « Year » : toutes les années sauf 2000, et uniquement pour les produits contenant la chaîne « bearing ».

Modificateurs d'ensemble associés à des opérateurs d'ensemble implicites

La manière standard d'écrire des sélections dans un modificateur d'ensemble consiste à utiliser un signe égal. Par exemple :

```
Year = {">2015"}
```

L'expression à droite du signe égal du modificateur d'ensemble est appelée ensemble d'éléments. Elle définit un ensemble de valeurs de champ distinctes, en d'autres termes, une sélection.

Cette notation définit une nouvelle sélection en ignorant la sélection active dans le champ. Ainsi, si l'identificateur d'ensemble contient une sélection dans ce champ, l'ancienne sélection sera remplacée par celle de l'ensemble d'éléments.

Si vous souhaitez baser votre sélection sur la sélection active dans le champ, vous devez utiliser une autre expression.

Par exemple, si vous souhaitez respecter l'ancienne sélection et ajouter la condition selon laquelle l'année doit figurer après 2015, vous pouvez écrire l'exemple suivant :

```
Year = Year * {">2015"}
```

L'astérisque est un opérateur d'ensemble qui définit une intersection. Vous obtiendrez donc l'intersection entre la sélection active dans `Year` et la condition supplémentaire selon laquelle l'année doit figurer après `2015`. Voici une autre manière de l'écrire :

```
Year *= {">2015"}
```

À savoir, l'opérateur d'affectation (`*=`) définit implicitement une intersection.

De même, il est possible de définir des unions, des exclusions et des différences symétriques implicites via : `+=`, `-=` et `/=`.

Exemples : Expressions de graphique pour modificateurs d'ensemble associés à des opérateurs d'ensemble implicites

Exemples - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Expressions de graphique associées à des opérateurs d'ensemble implicites

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

Dans une liste de pays, sélectionnez `Canada` et `Czech Republic`.

Tableau - Expressions de graphique associées à des opérateurs d'ensemble implicites

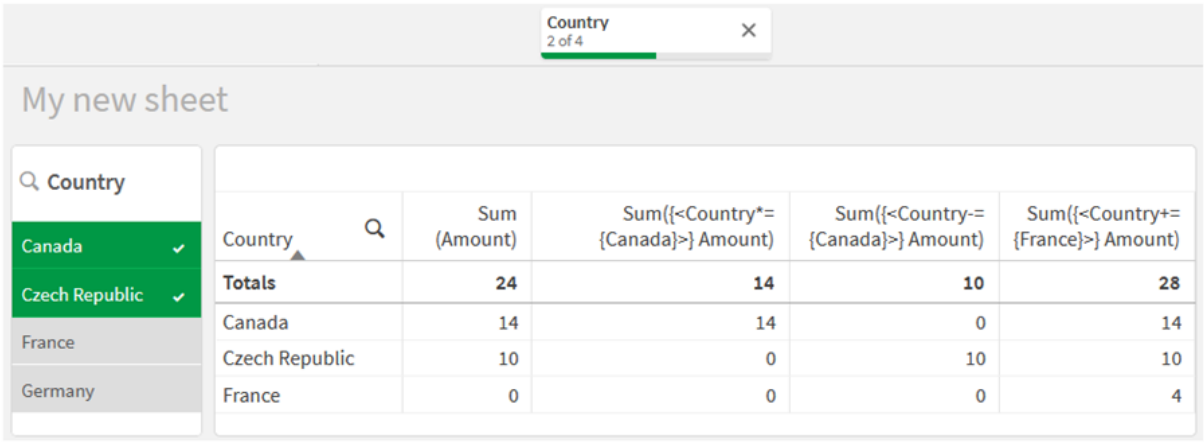
Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totals	**24**	**14**	**10**	**28**
Canada	14	14	0	14
Czech Republic	10	0	10	10
France	0	0	0	4

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    Somme `Amount` de la sélection active. Notez que seuls les pays `Canada` et `Czech Republic` ont des valeurs différentes de zéro.
  - `Sum({<Country*={Canada}>}Amount)`
    Somme `Amount` de la sélection active, intersectée avec la condition que `Country` soit `Canada`. Si `Canada` ne fait pas partie de la sélection de l'utilisateur, l'expression d'ensemble renvoie un ensemble vide et toutes les lignes de la colonne contiendront `0`.

- `Sum({<Country-={Canada}>}Amount)`
  Somme `Amount` de la sélection active, mais en commençant par exclure `Canada` de la sélection `Country`. Si `Canada` ne fait pas partie de la sélection de l'utilisateur, l'expression d'ensemble ne modifiera aucun chiffre.
- `Sum({<Country+={France}>}Amount)`
  Somme `Amount` de la sélection active, mais en commençant par ajouter `France` à la sélection `Country`. Si `France` fait déjà partie de la sélection de l'utilisateur, l'expression d'ensemble ne modifiera aucun chiffre.

Modificateurs d'ensemble associés à des opérateurs d'ensemble implicites



Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totals	**24**	**14**	**10**	**28**
Canada	14	14	0	14
Czech Republic	10	0	10	10
France	0	0	0	4

Exemples	Résultats
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Renvoie les ventes pour la sélection active en utilisant une union implicite afin d'ajouter les produits OurProduct1 et OurProduct2 à la liste des produits sélectionnés.
sum({$<Year += {“20*”,1997} - {2000} >} Sales)	Renvoie les ventes pour la sélection active en utilisant une union implicite afin d'ajouter un nombre d'années dans la sélection : 1997 et toutes celles qui commencent par 20 (sauf 2000). Vous noterez que si l'année 2000 fait partie de la sélection active, elle reste incluse après la modification. Équivaut à `<Year=Year + ({“20*”,1997}-{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Renvoie les ventes pour la sélection active, mais uniquement pour l'intersection des produits actuellement sélectionnés et du produit OurProduct1.

Modificateurs d'ensemble utilisant des fonctions d'ensemble

Parfois, vous devez définir un ensemble de valeurs de champ à l'aide d'une définition d'ensemble imbriquée. Par exemple, vous pouvez souhaiter sélectionner tous les clients qui ont acheté un produit spécifique sans sélectionner le produit.

Dans de tels cas, utilisez les fonctions d'ensemble d'éléments `P()` et `E()`. Elles renvoient les ensembles d'éléments de valeurs possibles et les valeurs exclues d'un champ, respectivement. À l'intérieur des parenthèses, vous pouvez spécifier le champ en question et une expression d'ensemble qui définit l'étendue. Par exemple :

```
P({1<Year = {2021}>} Customer)
```

Cela renverra l'ensemble des clients qui ont effectué des translations en 2021. Vous pouvez alors utiliser ce résultat dans un modificateur d'ensemble. Par exemple :

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Cette expression d'ensemble sélectionnera ces clients, mais ne limitera pas la sélection à 2021.

Il est impossible d'employer ces fonctions dans d'autres expressions.

De plus, seuls les ensembles naturels peuvent être utilisés à l'intérieur des fonctions d'ensemble d'éléments. Un ensemble naturel est un ensemble d'enregistrements qu'il est possible de définir par une sélection simple.

Par exemple, l'ensemble fourni par `{1-$}` ne peut pas toujours être défini par le biais d'une sélection et, de ce fait, il ne s'agit pas d'un ensemble naturel. L'application de ces fonctions à des ensembles non naturels renverra des résultats inattendus.

Exemples : Expressions de graphique pour modificateurs d'ensemble via des fonctions d'ensemble

Exemples - expressions de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Expressions de graphique

Créez une table dans une feuille Qlik Sense avec les expressions de graphique suivantes.

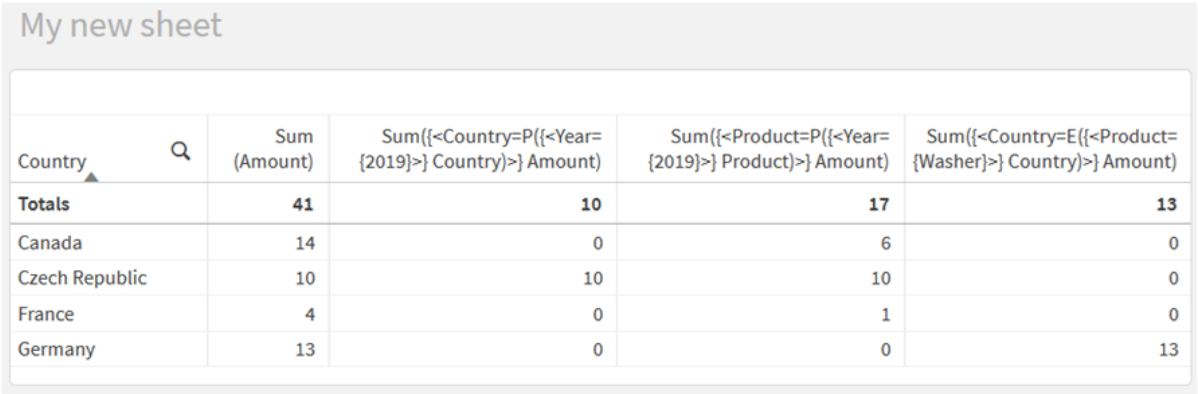
Tableau - Modificateurs d'ensemble utilisant des fonctions d'ensemble

Country	Sum (Amount)	Sum({<Country=P ({<Year= {2019}>}Country)>} Amount)	Sum({<Product=P ({<Year= {2019}>}Product)>} Amount)	Sum({<Country=E ({<Product= {Washer}>}Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Explication

- Dimensions :
  - `Country`
- Mesures :
  - `Sum(Amount)`
    somme `Amount` sans expression d'ensemble.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Somme `Amount` des pays associés à l'année `2019`. Cependant, cela ne limitera le calcul à l'année `2019`.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Somme `Amount` des produits associés à l'année `2019`. Cependant, cela ne limitera le calcul à l'année `2019`.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Somme `Amount` des pays non associés au produit `Washer`.

Modificateurs d'ensemble utilisant des fonctions d'ensemble



Exemples	Résultats
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Renvoie les ventes pour la sélection active, mais seuls les clients ayant un jour acheté le produit « Shoe » sont retenus. La fonction d'élément P( ) renvoie alors une liste de clients possibles, ceux qui sont concernés par la sélection « Shoe » dans le champ Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Comme précédemment. Si le champ est omis dans la fonction d'élément, la fonction renvoie les valeurs possibles pour le champ spécifié dans l'affectation extérieure.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Renvoie les ventes pour la sélection active, mais seulement les clients ayant un jour fourni le produit 'Shoe', à savoir, les clients également fournisseurs. La fonction d'élément P( ) renvoie alors une liste de fournisseurs possibles, ceux qui sont concernés par la sélection « Shoe » dans le champ Product. La liste des fournisseurs est alors utilisée comme sélection dans le champ Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Renvoie les ventes pour la sélection active, mais seuls les clients n'ayant jamais acheté le produit « Shoe » sont retenus. La fonction d'élément E( ) renvoie alors la liste des clients exclus, ceux qui ne font pas partie de la sélection « Shoe » dans le champ Product.

Expressions d'ensemble internes et externes

Les expressions d'ensemble peuvent être utilisées à l'intérieur et à l'extérieur des fonctions d'agrégation et sont encadrées par des accolades.

Lorsque vous utilisez une expression d'ensemble à l'intérieur d'une fonction d'agrégation, cela peut prendre la forme suivante :

Exemple : Expression d'ensemble interne

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

Utilisez une expression d'ensemble à l'extérieur de la fonction d'agrégation si vous avez des expressions avec plusieurs agrégations et si vous souhaitez éviter d'écrire la même expression d'ensemble dans chaque fonction d'agrégation.

Si vous utilisez une expression d'ensemble externe, elle doit être placée au début de l'étendue.

Exemple : Expression d'ensemble externe

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

Si vous utilisez une expression d'ensemble à l'extérieur de la fonction d'agrégation, vous pouvez également l'appliquer aux mesures principales existantes.

Exemple : Expression d'ensemble externe appliquée à une mesure principale

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

Une expression d'ensemble utilisée à l'extérieur des fonctions d'agrégation affecte l'expression toute entière, sauf si elle est placée entre parenthèses lorsque les parenthèses définissent l'étendue. Dans l'exemple de définition de l'étendue lexicale ci-dessous, l'expression d'ensemble est uniquement appliquée à l'agrégation à l'intérieur des parenthèses.

Exemple : Définition de l'étendue lexicale

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

Règles

Étendue lexicale

L'expression d'ensemble affecte l'expression toute entière, sauf si elle est placée entre parenthèses. Dans ce cas, les parenthèses définissent l'étendue lexicale.

Position

L'expression d'ensemble doit être placée au début de l'étendue lexicale.

Contexte

Le contexte est la sélection pertinente pour l'expression. En règle générale, le contexte est toujours l'état par défaut de la sélection active. En revanche, si un objet est défini sur un état alternatif, le contexte est l'état alternatif de la sélection active.

Vous pouvez également définir un contexte sous la forme d'une expression d'ensemble externe.

Héritage

Les expressions d'ensemble internes sont prioritaires sur les expressions d'ensemble externes. Si l'expression d'ensemble interne contient un identificateur d'ensemble, elle remplace le contexte. Sinon, le contexte et l'expression d'ensemble sont fusionnés.

- `{$<SetExpression>}` - remplace l'expression d'ensemble externe
- `{<SetExpression>}` - est fusionné avec l'expression d'ensemble externe

Affectation d'ensembles d'éléments

L'affectation d'ensembles d'éléments détermine le mode de fusion des deux sélections. Si un signe égal normal est utilisé, la sélection de l'expression d'ensemble interne est prioritaire. Sinon, l'opérateur d'ensemble implicite est utilisé.

- `{<Field={value}>}` - cette sélection interne remplace toute sélection externe dans “`Field`”.
- `{<Field+={value}>}` - cette sélection interne est fusionnée avec la sélection externe dans “`Field`”, via l'opérateur d'union.
- `{<Field*={value}>}` - cette sélection interne est fusionnée avec la sélection externe dans “`Field`”, via l'opérateur d'intersection.

Héritage en plusieurs étapes

L'héritage peut avoir lieu en plusieurs étapes. Exemples :

- Sélection active → `Sum(Amount)`
  La fonction d'agrégation utilisera le contexte, qui, ici, est la sélection active.
- Sélection active → `{<Set1>} Sum(Amount)`
  `Set1` héritera de la sélection active et le résultat sera le contexte pour la fonction d'agrégation.
- Sélection active → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` héritera de `Set1`, qui, à son tour, héritera de la sélection active, et le résultat sera le contexte pour la fonction d'agrégation.

Fonction Aggr()

La fonction `Aggr()` crée une agrégation imbriquée qui comporte deux agrégations indépendantes. Dans l'exemple ci-dessous, une fonction `Count()` est calculée pour chaque valeur de `Dim`, et le tableau obtenu est agrégé via la fonction `Sum()`.

Exemple :

`Sum(Aggr(Count(X),Dim))`
`Count()` est l'agrégation interne et `Sum()` l'agrégation externe.

- L'agrégation interne n'hérite d'aucun contexte de l'agrégation externe.
- L'agrégation interne hérite du contexte de la fonction `Aggr()`, qui contient une expression d'ensemble.
- La fonction `Aggr()` et la fonction d'agrégation externe héritent toutes les deux du contexte d'une expression d'ensemble externe.

Didacticiel - Création d'une expression d'ensemble

Vous pouvez créer des expressions d'ensemble pour prendre en charge l'analyse de données. Dans ce contexte, l'analyse est souvent appelée analyse d'ensembles. L'analyse d'ensembles est une manière de définir une étendue différente de l'ensemble d'enregistrements défini par la sélection active dans une carte.

Ce que vous allez apprendre

Ce didacticiel fournit les données et les expressions de graphique permettant de créer des expressions d'ensemble via des modificateurs, des identificateurs et des opérateurs d'ensemble.

À qui s'adresse ce didacticiel

Ce didacticiel s'adresse aux développeurs d'applications qui ont l'habitude d'utiliser l'éditeur de script et les expressions de graphique.

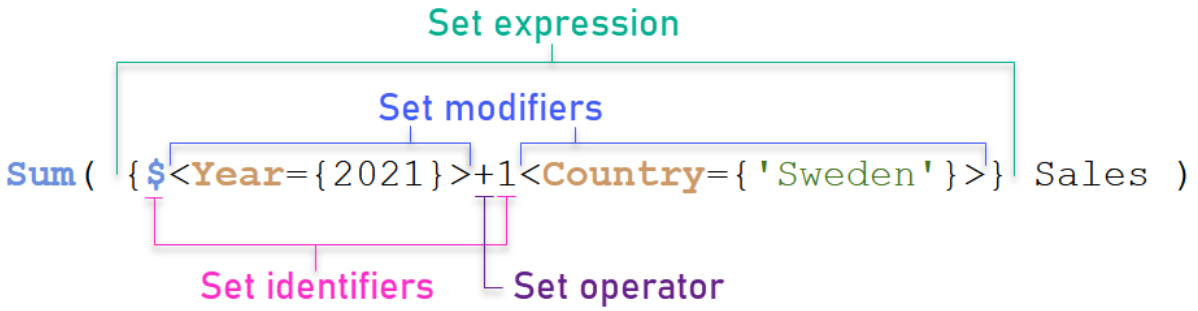
Ce que vous devez faire avant de commencer

L'affectation d'un accès Professional Qlik Sense Enterprise vous permettant de charger des données et de créer des applications.

Éléments d'une expression d'ensemble

Les expressions d'ensemble sont insérées dans une fonction d'agrégation telle que `Sum()`, `Max()`, `Min()`, `Avg()` ou `Count()`. Les expressions d'ensemble sont créées à partir de blocs de construction appelés éléments. Ces éléments sont des modificateurs, des identificateurs et des opérateurs d'ensemble.

Éléments d'une expression d'ensemble



L'expression d'ensemble ci-dessus, par exemple, est créée à partir de l'agrégation `Sum(Sales)`. L'expression d'ensemble est encadrée par des accolades externes : `{ }`

Le premier opérande de l'expression est le suivant : `$<Year={2021}>`

Cet opérande renvoie les ventes pour l'année `2021` pour la sélection active. Le modificateur, `<Year={2021}>`, contient la sélection de l'année 2021. L'identificateur d'ensemble `$` indique que l'expression d'ensemble est basée sur la sélection active.

Le deuxième opérande de l'expression est le suivant : `1<Country={'Sweden'}>`

Cet opérande renvoie Sales pour Sweden. Le modificateur, `<Country={'Sweden'}>`, contient la sélection du pays Sweden. L'identificateur d'ensemble `1` indique que les sélections effectuées dans l'application seront ignorées.

Pour finir, l'opérateur d'ensemble `+` indique que l'expression renvoie un ensemble composé des enregistrements qui appartiennent à l'un ou l'autre des deux opérandes d'ensemble.

Didacticiel Création d'une expression d'ensemble

Pour créer les expressions d'ensemble montrées dans ce didacticiel, suivez les procédures suivantes.

Création d'une application et chargement de données

Procédez comme suit :

1. Permet de créer une nouvelle application.
2. Cliquez sur **Éditeur de script**. Sinon, cliquez sur **Préparer** > **Éditeur de chargement de données** dans la barre de navigation.
3. Créez une section dans l'**éditeur de chargement de données**.

4. Copiez les données suivantes et collez-les dans la nouvelle section : *Données du didacticiel sur les expressions d'ensemble (page 318)*
5. Cliquez sur **Charger les données**. Les données sont chargées sous forme de chargement inline.

Création d'expressions d'ensemble avec des modificateurs

Le modificateur d'ensemble se compose d'un ou de plusieurs noms de champ, chacun suivi d'une sélection devant être effectuée dans le champ. Le modificateur est encadré par des crochets angulaires. Par exemple, dans cette expression d'ensemble :

```
Sum ( {<Year = {2015}>} Sales )
```

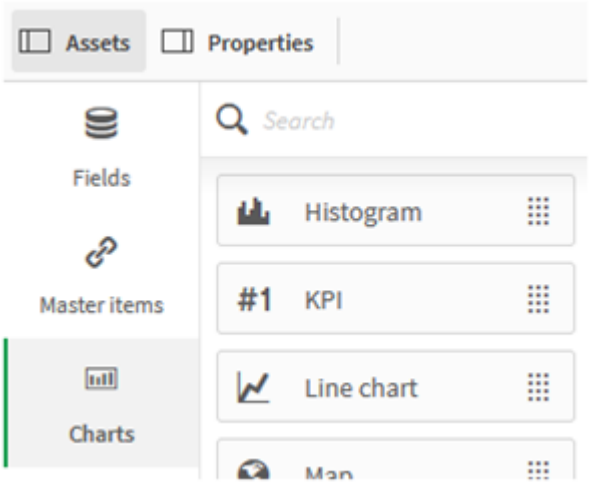
Le modificateur est :

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

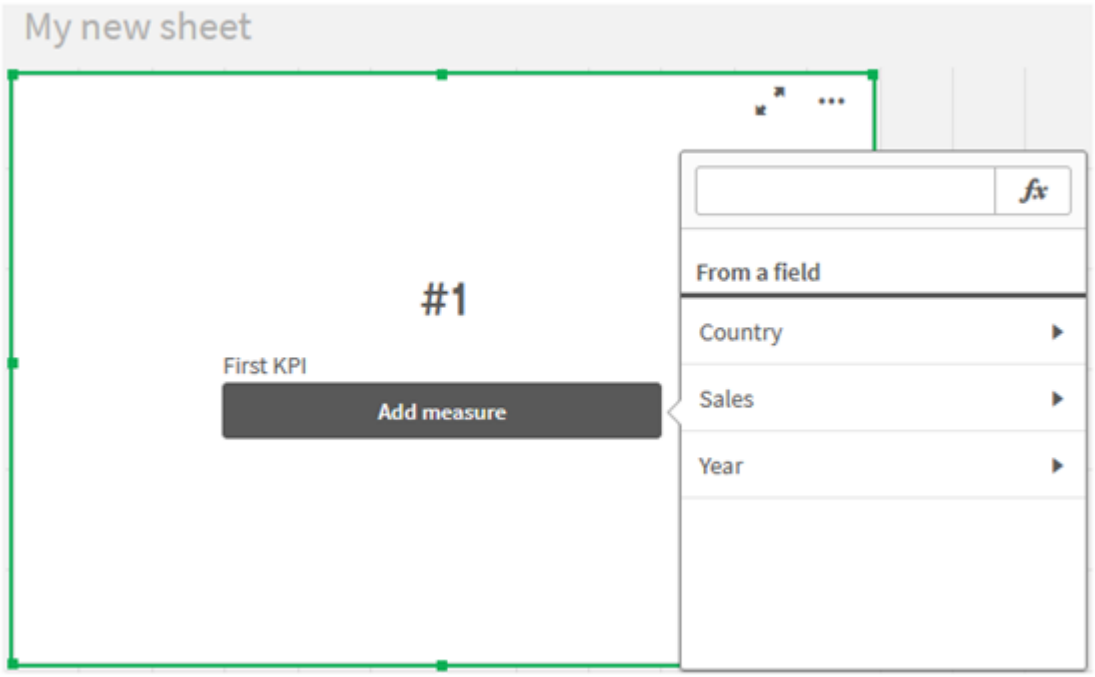
Ce modificateur spécifie que les données de l'année 2015 seront sélectionnées. Les accolades encadrant le modificateur indiquent une expression d'ensemble.

Procédez comme suit :

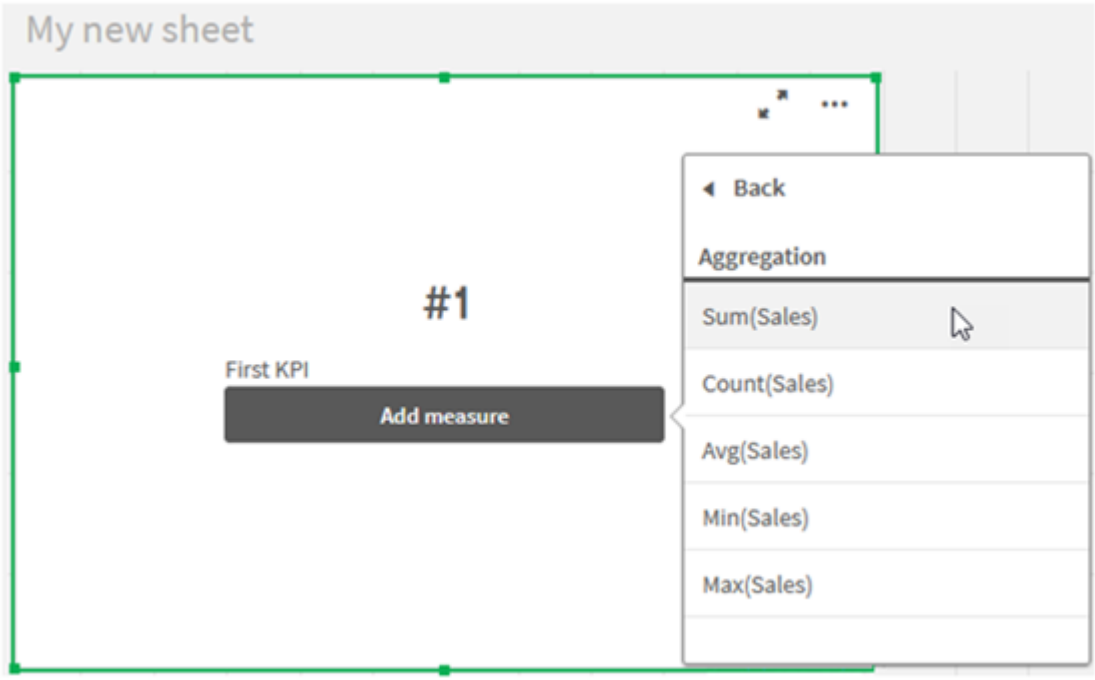
1. Dans une feuille, ouvrez le panneau des **Ressources** dans la barre de navigation et cliquez sur **Graphiques**.



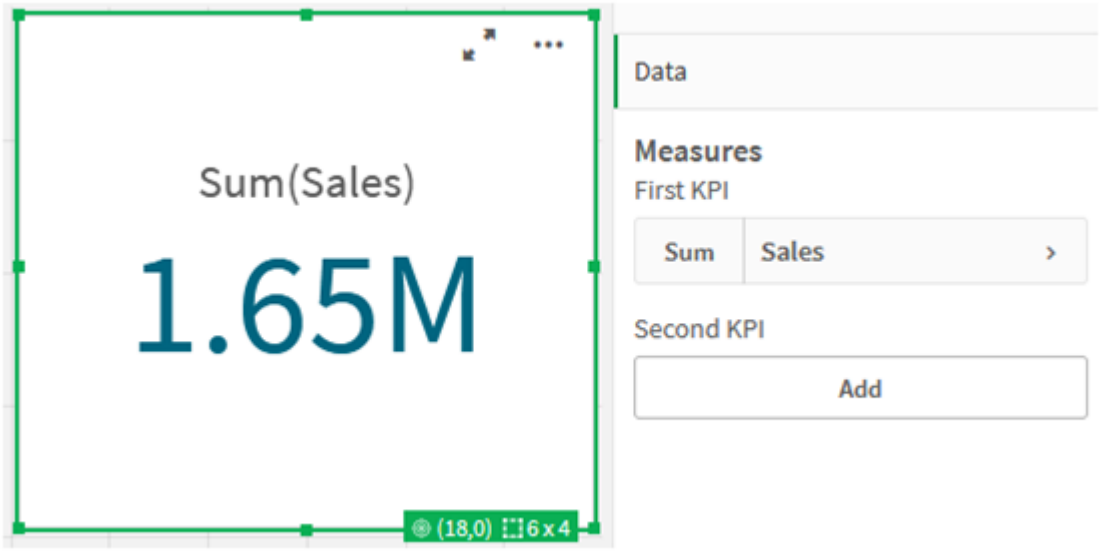
2. Glissez un indicateur **KPI** sur la feuille, puis cliquez sur **Ajouter une mesure**.



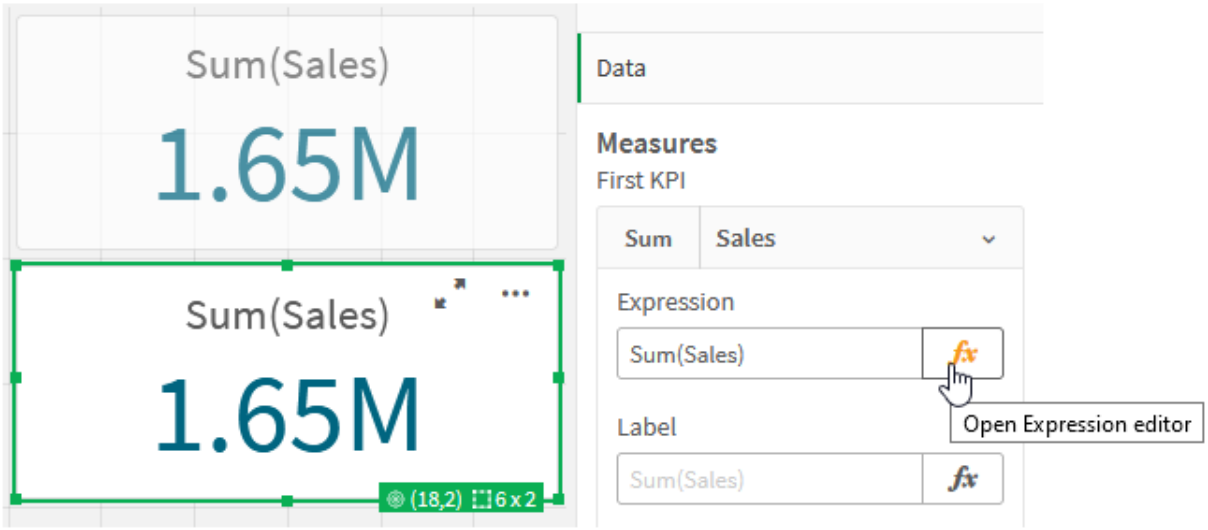
3. Cliquez sur `Sales`, puis sélectionnez `Sum(Sales)` pour l'agrégation.



L'indicateur KPI affiche la somme des ventes pour toutes les années.

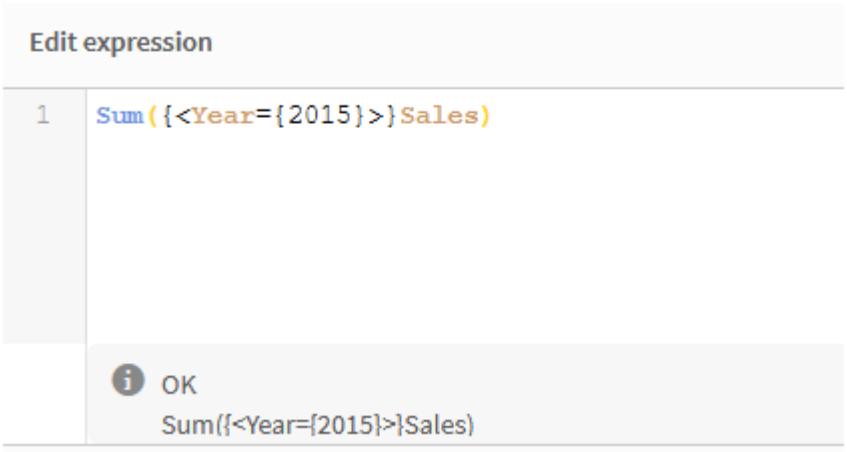


4. Copiez et collez l'indicateur KPI pour créer un nouvel indicateur KPI.
5. Cliquez sur le nouvel indicateur KPI, sur **Sales** sous **Mesures**, puis sur **Ouvrir l'éditeur d'expression**.

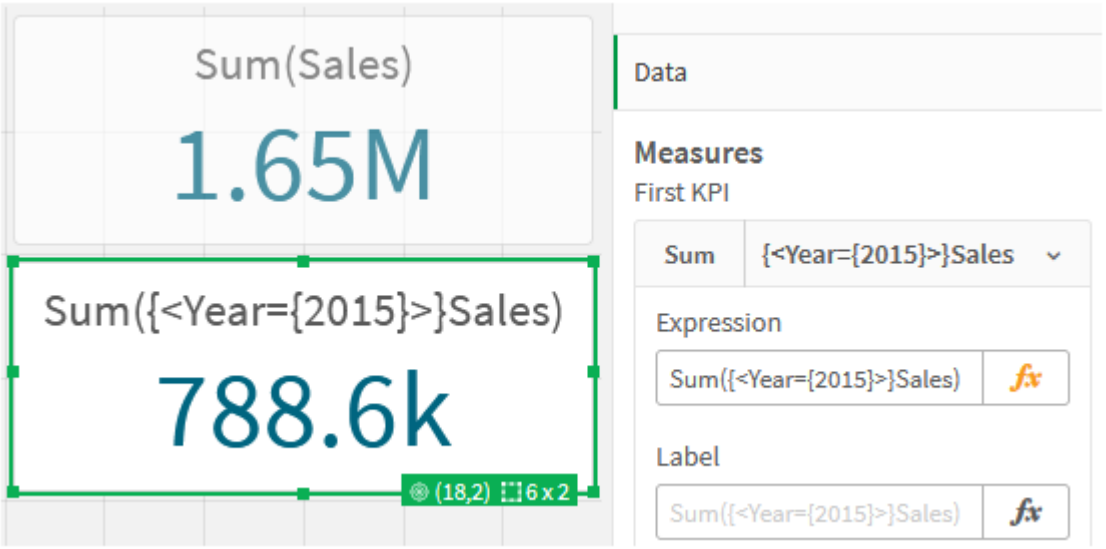


L'éditeur d'expression s'ouvre avec l'agrégation `Sum(Sales)`.

6. Dans l'éditeur d'expression, créez une expression pour additionner Sales uniquement pour 2015 :
   i. Ajoutez des accolades pour indiquer une expression d'ensemble : `Sum({}Sales)`
   i. Ajoutez des crochets angulaires pour indiquer un modificateur d'ensemble : `Sum({<>}Sales)`
   ii. Dans les crochets angulaires, ajoutez le champ à sélectionner, dans ce cas, le champ `Year`, suivi d'un signe égal. Ensuite, encadrez 2015 à l'aide d'une autre paire d'accolades. Le modificateur d'ensemble obtenu est le suivant : `{<Year={2015}>}`.
   L'expression complète est la suivante :
   `Sum({<Year={2015}>}Sales)`

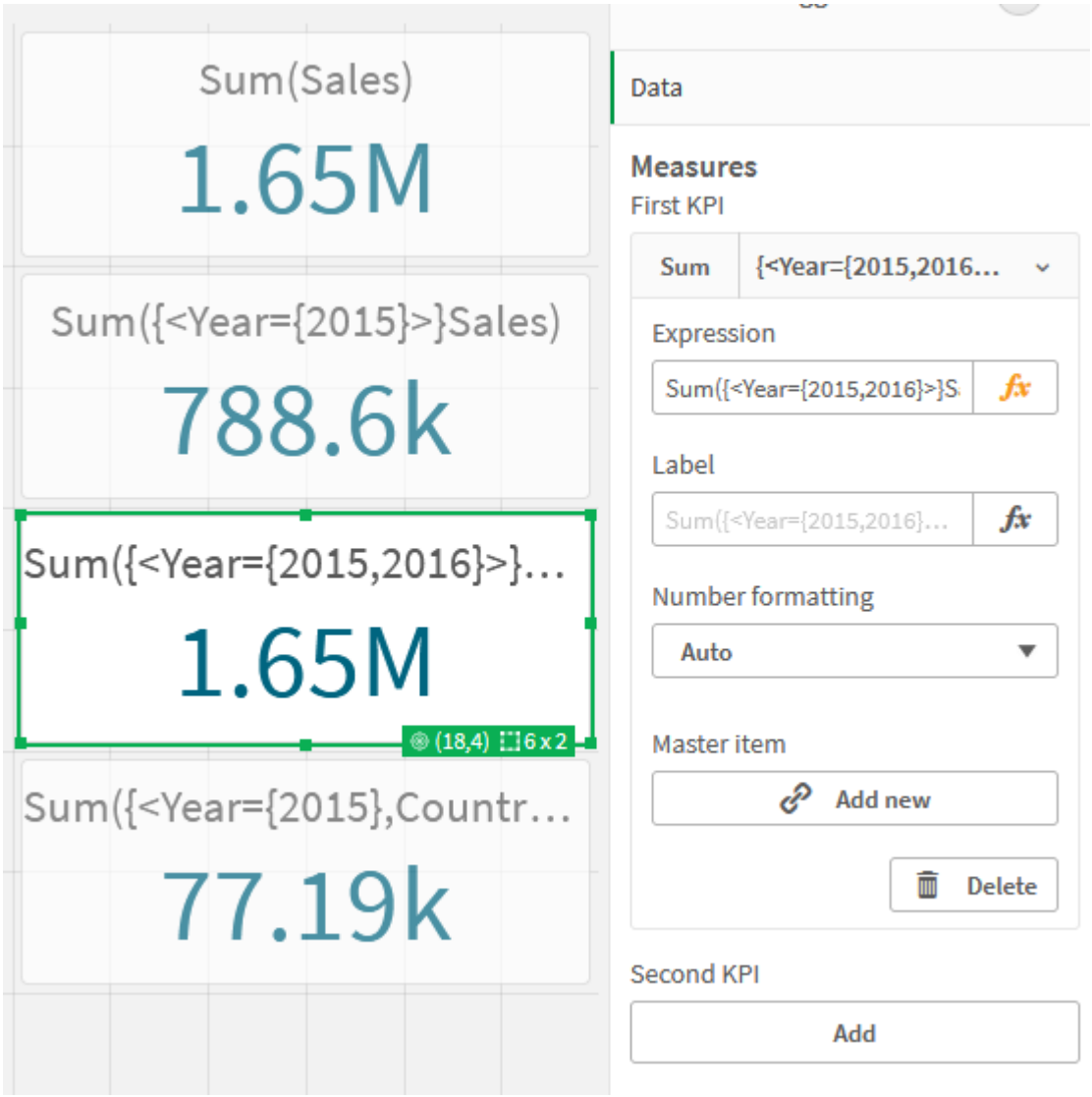


   iii. Cliquez sur **Appliquer** pour enregistrer l'expression et fermer l'éditeur d'expression. La somme de Sales pour 2015 apparaît dans l'indicateur KPI.



7. Créez deux autres indicateurs KPI avec les expressions suivantes :
   `Sum({<Year={2015,2016}>}Sales)`
   Le modificateur du résultat ci-dessus est `<Year={2015,2016}>`. L'expression renverra la somme de Sales pour 2015 et 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Le modificateur du résultat ci-dessus est `<Year={2015}, Country={'Germany'}>`. L'expression renverra la somme de Sales pour 2015, où 2015 s'intersecte avec Germany.

Indicateurs KPI via des modificateurs d'ensemble



Ajout d'identificateurs d'ensemble

Les expressions d'ensemble ci-dessus utilisent les sélections actives comme base, parce qu'aucun identificateur n'a été utilisé. Ensuite, ajoutez des identificateurs pour spécifier le comportement lorsque des sélections sont effectuées.

Procédez comme suit :

Sur la feuille, créez ou copiez les expressions d'ensemble suivantes :

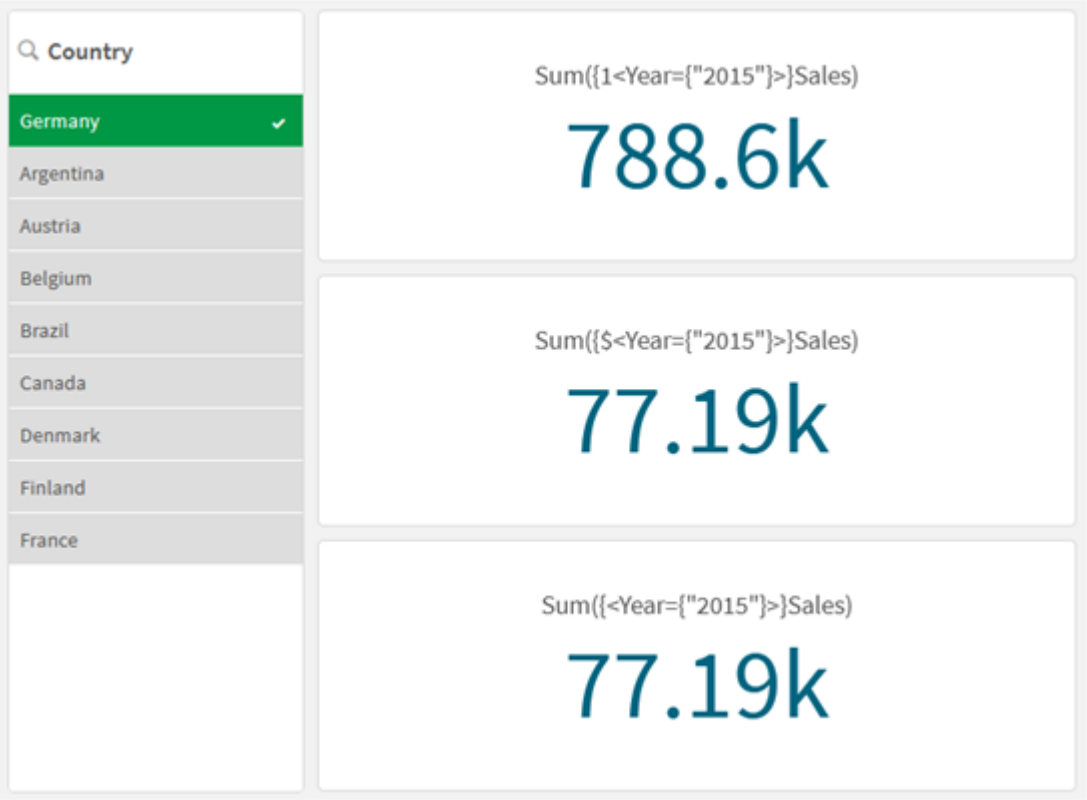
```
Sum({$<Year={"2015"}>}Sales
```

L'identificateur `$` basera l'expression d'ensemble sur les sélections actives effectuées dans les données. En l'absence d'identificateur, il s'agit également du comportement par défaut.

```
Sum({1<Year={"2015"}>}Sales)
```

L'identificateur `1` amènera l'agrégation de `Sum(Sales)` sur 2015 à ignorer la sélection active. La valeur de l'agrégation ne changera pas lorsque l'utilisateur effectuera d'autres sélections. Par exemple, lorsque Germany est sélectionné ci-dessous, la valeur de la somme agrégée de 2015 ne change pas.

Indicateurs KPI via des identificateurs et des modificateurs d'ensemble



Ajout d'opérateurs

Les opérateurs d'ensemble permettent d'inclure, d'exclure ou d'intersecter des ensembles de données. Tous les opérateurs utilisent les ensembles comme opérandes et renvoient un ensemble pour résultat.

Vous pouvez utiliser des opérateurs d'ensemble dans deux situations différentes :

- Pour effectuer une opération d'ensemble sur des identificateurs d'ensemble représentant des ensembles d'enregistrements dans des données.
- Pour effectuer une opération d'ensemble sur les ensembles d'éléments, sur les valeurs de champ ou à l'intérieur d'un modificateur d'ensemble.

Procédez comme suit :

Sur la feuille, créez ou copiez l'expression d'ensemble suivante :

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

L'opérateur de signe plus `(+)` produit une union des ensembles de données pour 2015 et Germany. Comme expliqué avec les identificateurs d'ensemble ci-dessus, l'identificateur de signe dollar `($)` signifie que les sélections actives seront utilisées pour le premier opérande, `<Year={2015}>`, et qu'elles seront respectées. L'identificateur `1` signifie que la sélection sera ignorée pour le deuxième opérande, `<Country={'Germany'}>`.

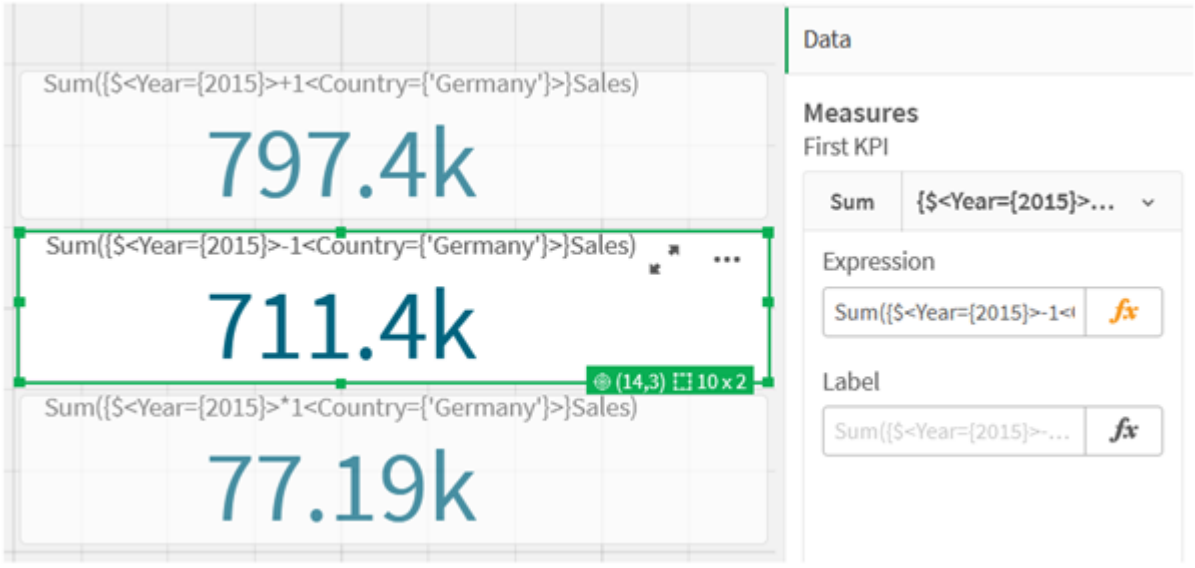
Indicateur KPI avec l'opérateur de signe plus (+)

Sinon, utilisez un signe moins `(-)` pour renvoyer un ensemble de données constitué des enregistrements appartenant à 2015, mais pas à Germany. Ou utilisez un astérisque `(*)` pour renvoyer un ensemble constitué des enregistrements qui appartiennent aux deux ensembles.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

Indicateurs KPI utilisant des opérateurs



Données du didacticiel sur les expressions d'ensemble

Script de chargement

Chargez les données suivantes sous forme de chargement inline, puis créez les expressions de graphique du didacticiel.

```
//Create table SalesByCountry
SalesByCountry:
Load * Inline [
Country, Year, Sales
Argentina, 2016, 66295.03
Argentina, 2015, 140037.89
Austria, 2016, 54166.09
Austria, 2015, 182739.87
Belgium, 2016, 182766.87
Belgium, 2015, 178042.33
Brazil, 2016, 174492.67
Brazil, 2015, 2104.22
Canada, 2016, 101801.33
Canada, 2015, 40288.25
Denmark, 2016, 45273.25
Denmark, 2015, 106938.41
Finland, 2016, 107565.55
Finland, 2015, 30583.44
France, 2016, 115644.26
France, 2015, 30696.98
Germany, 2016, 8775.18
Germany, 2015, 77185.68
];
```

Syntaxe des expressions d'ensemble

La syntaxe complète (à l'exclusion de l'utilisation facultative des accolades classiques pour définir l'ordre de priorité) est décrite à l'aide du code BNF (Backus-Naur Formalism) :

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

3.3 Syntaxe générale pour les expressions de graphique

La structure de syntaxe générale suivante peut être utilisée pour les expressions de graphique, avec de nombreux paramètres facultatifs :

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

où :

constant est une chaîne (texte, date ou heure) placée entre guillemets simples ou un nombre. Les constantes sont écrites sans séparateur de milliers et avec un point comme séparateur décimal.

expressionname est le nom (l'étiquette) d'une autre expression figurant dans le même graphique.

operator1 est un opérateur unaire (qui agit sur une seule expression, celle qui se trouve à droite).

operator2 est un opérateur binaire (qui agit sur deux expressions, une de chaque côté).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```

Le nombre et les types de paramètres ne sont pas arbitraires. Ils dépendent de la fonction utilisée.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```

Le nombre et les types de paramètres ne sont pas arbitraires. Ils dépendent de la fonction utilisée.

3.4 Syntaxe générale pour les agrégations

La structure de syntaxe générale suivante peut être utilisée pour les agrégations, avec de nombreux paramètres facultatifs :

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```

fieldref désigne un nom de champ.

```
functionaggr ::= functionname ( parameters2 )
```

Les expressions et les fonctions peuvent ainsi être imbriquées librement : du moment que le nom de champ **fieldref** est toujours inclus dans exactement une fonction d'agrégation et que l'expression renvoie une valeur interprétable, Qlik Sense n'affiche pas de messages d'erreur.

4 Opérateurs

Cette section décrit les opérateurs pouvant être utilisés dans Qlik Sense. Il en existe deux types d'opérateurs :

- Opérateurs unaires (qui n'admettent qu'un seul opérande)
- Opérateurs binaires (qui admettent deux opérandes)

La plupart des opérateurs sont binaires.

Il est possible de définir les opérateurs suivants :

- Opérateurs de bits
- Opérateurs logiques
- Opérateurs mathématiques
- Opérateurs relationnels
- Opérateurs de chaîne

4.1 Opérateurs de bits

Tous les opérateurs de bits convertissent (tronquent) les opérandes en entiers signés (32 bits) et renvoient le résultat de la même façon. Toutes les opérations sont effectuées bit par bit. S'il est impossible d'interpréter un opérande sous forme de nombre, l'opération renvoie la valeur NULL.

Opérateurs de bits

Opérateur	Nom complet	Description
bitnot	Inverse de bits.	Opérateur unaire. L'opération renvoie l'inverse logique de l'opérande exécuté bit par bit. **Exemple :** `bitnot 17` renvoie -18.
bitand	Et binaire.	L'opération renvoie le ET logique des opérandes exécutés bit par bit. **Exemple :** `17 bitand 7` renvoie 1.
bitor	Ou binaire.	L'opération renvoie le OU logique des opérandes exécutés bit par bit. **Exemple :** `17 bitor 7` renvoie 23.

Opérateur	Nom complet	Description
bitxor	Ou exclusif binaire.	L'opération renvoie le Ou logique exclusif des opérandes exécutés bit par bit. **Exemple :** `17 bitxor 7` renvoie 22.
>>	Décalage de bit à droite.	L'opération renvoie le premier opérande décalé vers la droite. Le nombre d'étapes est défini dans le deuxième opérande. **Exemple :** `8 >> 2` renvoie 2.
<<	Décalage de bit à gauche.	L'opération renvoie le premier opérande décalé vers la gauche. Le nombre d'étapes est défini dans le deuxième opérande. **Exemple :** `8 << 2` renvoie 32.

4.2 Opérateurs logiques

Tous les opérateurs logiques interprètent les opérandes logiquement et renvoient True (-1) ou False (0).

Opérateurs logiques

Opérateur	Description
not	Inverse logique. L'un des opérateurs unaires. L'opération renvoie l'inverse logique de l'opérande.
and	Et logique. L'opération renvoie le et logique des opérandes.
or	Ou logique. L'opération renvoie le Ou logique des opérandes.
Xor	Ou logique exclusif. L'opération renvoie le ou logique exclusif des opérandes. Cela s'apparente au Ou logique à cette différence près que le résultat correspond à False si les deux opérandes sont définis sur True.

4.3 Opérateurs mathématiques

Tous les opérateurs mathématiques utilisent les valeurs numériques des opérandes et renvoient une valeur numérique.

Opérateurs mathématiques

Opérateur	Description
+	Signe d'un nombre positif (opérateur unaire) ou d'une addition arithmétique. L'opération binaire renvoie la somme des deux opérandes.
-	Signe d'un nombre négatif (opérateur unaire) ou d'une soustraction arithmétique. L'opération unaire renvoie l'opérande multiplié par -1 tandis que l'opération binaire renvoie la différence entre les deux opérandes.
*	Multiplication arithmétique. L'opération renvoie le produit des deux opérandes.
/	Division arithmétique. L'opération renvoie le rapport des deux opérandes.

4.4 Opérateurs relationnels

Tous les opérateurs relationnels comparent les valeurs des opérandes et renvoient True (-1) ou False (0) comme résultat. Tous les opérateurs relationnels sont binaires.

Opérateurs relationnels

Opérateur	Description
<	Inférieur à. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.
<=	Inférieur ou égal à. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.
>	Supérieur à. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.
>=	Supérieur ou égal à. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.
=	Égal à. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.
<>	Différent de. Effectue une comparaison numérique si les deux opérandes peuvent être interprétés numériquement. L'opération renvoie le résultat logique de l'évaluation de la comparaison.

Opérateur	Description
precedes	Contrairement à l'opérateur **<**, cet opérateur ne tente pas de réaliser une interprétation numérique des valeurs d'argument avant d'effectuer la comparaison. L'opération renvoie true si la valeur située à gauche de l'opérateur a une représentation textuelle qui, lors de la comparaison de chaînes, est antérieure à la représentation textuelle de la valeur de droite. **Exemple :** `'1 ' precedes ' 2'` renvoie FALSE. `' 1' precedes ' 2'` renvoie TRUE. Car la valeur ASCII d'un espace (' ') a une valeur moindre que la valeur ASCII d'un nombre. Comparez cela à l'exemple suivant : `'1 ' < ' 2'` renvoie TRUE `' 1' < ' 2'` renvoie TRUE.
follows	Contrairement à l'opérateur **>**, cet opérateur ne tente pas de réaliser une interprétation numérique des valeurs d'argument avant d'effectuer la comparaison. L'opération renvoie true si la valeur située à gauche de l'opérateur a une représentation textuelle qui, lors de la comparaison de chaînes, est postérieure à la représentation textuelle de la valeur de droite. **Exemple :** `' 2' follows '1'` renvoie FALSE ' 2' follows ' 1' renvoie TRUE. Car la valeur ASCII d'un espace (' ') a une valeur moindre que la valeur ASCII d'un nombre. Comparez cela à l'exemple suivant : `' 2' > ' 1'` renvoie TRUE. `' 2' > '1 '` renvoie TRUE.

4.5 Opérateurs de chaîne

Il existe deux opérateurs de chaîne. Le premier utilise les valeurs de chaîne des opérandes et renvoie une chaîne comme résultat. L'autre compare les opérandes et renvoie une valeur booléenne pour indiquer qu'une correspondance existe.

&

Concaténation de chaînes. L'opération renvoie une chaîne textuelle composée de deux chaînes d'opérandes, placées l'une après l'autre.

Exemple :

`'abc' & 'xyz'` renvoie 'abcxyz'.

like

Comparaison de chaînes contenant des caractères génériques. L'opération renvoie la valeur booléenne True (-1) si la chaîne qui précède l'opérateur correspond à la chaîne qui le suit. La deuxième chaîne peut contenir les caractères génériques * (tout nombre parmi de caractères arbitraires) ou ? (un caractère arbitraire).

Exemple :

`'abc' like 'a*'` renvoie True (-1).

`'abcd' like 'a?c*'` renvoie True (-1).

`'abc' like 'a??bc'` renvoie False (0).

5 Fonctions de script et de graphique

Transformez et agrégez les données à l'aide de fonctions dans des scripts de chargement de données et des expressions de graphique.

De nombreuses fonctions s'emploient de la même manière dans les scripts de chargement de données et les expressions de graphique, aux exceptions près suivantes :

- Certaines fonctions s'utilisent exclusivement dans les scripts de chargement de données, auquel cas elles sont signalées par l'indication – fonction de script.
- Certaines fonctions s'utilisent exclusivement dans les expressions de graphique, auquel cas elles sont signalées par l'indication – fonction de graphique.
- D'autres fonctions encore s'utilisent à la fois dans les scripts de chargement de données et les expressions de graphique, mais en présentant des différences au niveau de leurs paramètres et de leur application. Elles font l'objet d'une description dans des rubriques distinctes, signalées par l'indication – fonction de script ou – fonction de graphique.

5.1 Connexions analytiques pour les extensions côté serveur (SSE)

Pour afficher les fonctions des connexions analytiques, vous devez configurer ces dernières et préalablement lancer Qlik Sense.

Vous devez utiliser QMC pour configurer les connexions analytiques (voir la rubrique « Creating an analytic connection » du guide Manage Qlik Sense sites).

Dans Qlik Sense Desktop, vous configurez les connexions analytiques en éditant le fichier *Settings.ini* (voir la rubrique « Configuring analytic connections in Qlik Sense Desktop » du guide Qlik Sense Desktop).

5.2 Fonctions d'agrégation

La famille de fonctions appelée fonctions d'agrégation se compose de fonctions qui utilisent plusieurs valeurs de champ comme entrée et qui renvoient un seul résultat par groupe, où le regroupement est défini par une dimension de graphique ou une clause **group by** dans l'instruction de script.

Les fonctions d'agrégation comprennent, entre autres, **Sum()**, **Count()**, **Min()** et **Max()**.

La plupart des fonctions d'agrégation s'utilisent à la fois dans le script de chargement de données et dans les expressions de graphique, même si leur syntaxe diffère.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Lors du nommage d'une entité, évitez d'attribuer le même nom à plus d'un champ, d'une variable ou d'une mesure. Il existe un ordre de précédence strict pour la résolution des conflits entre les entités portant des noms identiques. Cet ordre est reflété dans tous les objets ou contextes dans lesquels ces entités sont utilisées. Cet ordre des priorités est le suivant :

- À l'intérieur d'une agrégation, un champ est prioritaire sur une variable. Les étiquettes de mesure n'ont pas d'importance dans les agrégations et ne sont pas priorisées.
- En dehors d'une agrégation, une étiquette de mesure est prioritaire sur une variable, qui, à son tour, est prioritaire sur un nom de champ.
- De plus, en dehors d'une agrégation, une mesure peut être réutilisée en référençant son étiquette, sauf si l'étiquette est en fait une étiquette calculée. Dans ce cas, la mesure perd en signification afin de réduire le risque d'auto-référence et le nom sera toujours interprété d'abord comme une étiquette de mesure, puis comme un nom de champ, et enfin comme un nom de variable.

Utilisation des fonctions d'agrégation dans un script de chargement de données

Les fonctions d'agrégation peuvent uniquement être utilisées dans des instructions **LOAD** et **SELECT**.

Utilisation des fonctions d'agrégation dans les expressions de graphique

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Une fonction d'agrégation regroupe l'ensemble des enregistrements possibles définis par la sélection. Il est toutefois possible de définir un ensemble alternatif d'enregistrements en utilisant une expression d'ensemble dans une analyse d'ensembles.

Mode de calcul des agrégations

Une agrégation effectue une boucle sur les enregistrements d'une table spécifique, agrégeant ainsi les enregistrements qu'elle contient. Par exemple, **Count**(<Field>) compte le nombre d'enregistrements de la table dans laquelle réside <Field>. Si vous souhaitez agréger uniquement les valeurs de champ distinctes, vous devez utiliser la clause **distinct** comme suit : **Count**(**distinct** <Field>).

Si la fonction d'agrégation contient des champs provenant de différentes tables, elle effectue une boucle sur les enregistrements du produit croisé des tables des champs constitutifs. Cela affecte les performances. C'est pourquoi il est recommandé d'éviter de telles agrégations, en particulier lorsque vous avez de grandes quantités de données.

Agrégation de champs clés

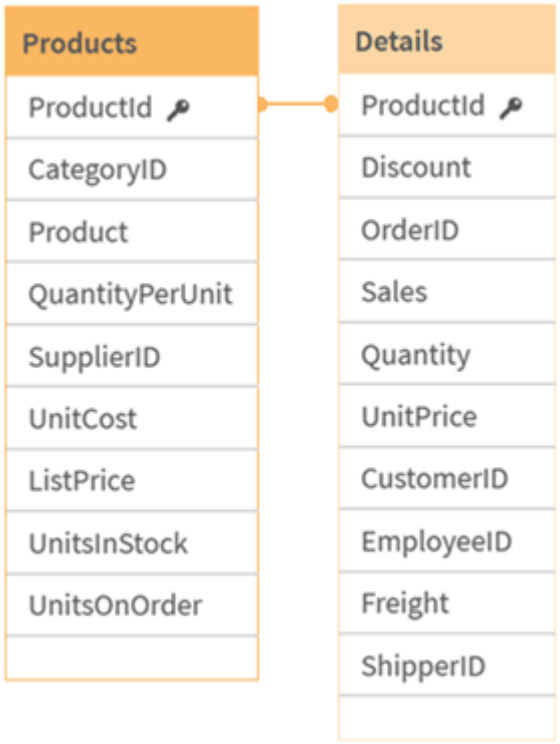
Le mode de calcul des agrégations signifie que vous ne pouvez pas agréger des champs clés, car il est difficile de savoir quelle table utiliser pour l'agrégation. Par exemple, si le champ <Key> relie deux tables, il est difficile de savoir si **Count**(<Key>) doit renvoyer le nombre d'enregistrements de la première ou de la deuxième table.

En revanche, si vous utilisez la clause **distinct**, l'agrégation est bien définie et peut être calculée.

C'est pourquoi, si vous utilisez un champ clé à l'intérieur d'une fonction d'agrégation sans la clause **distinct**, Qlik Sense renverra un nombre qui risque de n'avoir aucun sens. La solution consiste à utiliser la clause **distinct** ou une copie de la clé – une copie qui réside dans une seule table.

Par exemple, dans les tables suivantes, ProductID est la clé entre les tables.

Clé ProductID entre tables Produits et Détails



Count(ProductID) peut être compté soit dans la table Products (qui contient un seul enregistrement par produit – ProductID est la clé primaire), soit dans la table Details (qui contient très probablement plusieurs enregistrements par produit). Si vous souhaitez compter le nombre de produits distincts, vous devez utiliser Count(distinct ProductID). Si vous souhaitez compter le nombre de lignes d'une table spécifique, vous ne devez pas utiliser la clé.

Fonctions d'agrégation de base

Vue d'ensemble des fonctions d'agrégation de base

Les fonctions d'agrégation de base correspondent au groupe des fonctions d'agrégation les plus courantes.

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Fonctions d'agrégation de base utilisées dans le script de chargement de données

FirstSortedValue

FirstSortedValue() renvoie la valeur de l'expression spécifiée dans **value** qui correspond au résultat du tri de l'argument **sort_weight**, par exemple, le nom du produit ayant le prix unitaire le plus bas. Il est possible de spécifier la nième valeur dans l'ordre de tri dans l'argument **rank**. Si plusieurs valeurs résultantes partagent le même champ **sort_weight** pour la fonction **rank** spécifiée, la fonction renvoie la valeur NULL. Les valeurs sont itérées sur un nombre d'enregistrements définis par une clause **group by** ou agrégées sur l'ensemble de données entier si aucune clause **group by** n'a été définie.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() permet de déterminer la valeur numérique la plus élevée contenue dans les données agrégées de l'expression définie par une clause **group by**. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus élevée.

```
Max ( expression[, rank])
```

Min

Min() renvoie la valeur numérique la plus basse contenue dans les données agrégées de l'expression définie par une clause **group by**. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus basse.

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

Mode

Mode() renvoie la valeur la plus fréquente, la valeur de mode, contenue dans les données agrégées de l'expression définie par une clause **group by**. La fonction **Mode()** peut renvoyer aussi bien des valeurs numériques que des valeurs textuelles.

```
Mode (expression )
```

Only

Only() renvoie une valeur s'il n'y a qu'un seul résultat possible dans les données agrégées. Si les enregistrements contiennent une seule valeur, c'est elle qui est renvoyée, sinon c'est la valeur NULL. Utilisez la clause **group by** si vous souhaitez que l'évaluation porte sur plusieurs enregistrements. La fonction **Only()** peut renvoyer des valeurs numériques comme des valeurs textuelles.

```
Only (expression )
```

Sum

Sum() calcule le nombre total de valeurs agrégées dans l'expression définie par une clause **group by**.

```
Sum ([distinct]expression)
```

Fonctions d'agrégation de base utilisées dans les expressions de graphique

Les fonctions d'agrégation dans les graphiques s'appliquent uniquement à des champs figurant dans des expressions de graphique. L'expression de l'argument d'une fonction d'agrégation ne doit contenir aucune autre fonction d'agrégation.

FirstSortedValue

FirstSortedValue() renvoie la valeur de l'expression spécifiée dans **value** qui correspond au résultat du tri de l'argument **sort_weight**, par exemple, le nom du produit ayant le prix unitaire le plus bas. Il est possible de spécifier la nième valeur dans l'ordre de tri dans l'argument **rank**. Si plusieurs valeurs résultantes partagent le même champ **sort_weight** pour la fonction **rank** spécifiée, la fonction renvoie la valeur NULL.

```
FirstSortedValue - fonction de graphique([{SetExpression}] [DISTINCT] [TOTAL
[<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max() permet de déterminer la valeur la plus élevée parmi les données agrégées. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus élevée.

```
Max - fonction de graphiqueMax() permet de déterminer la valeur la plus
élevée parmi les données agrégées. Si vous spécifiez un argument rank n, vous
pouvez rechercher la énième valeur la plus élevée. Il est également
recommandé de consulter la description des fonctions FirstSortedValue et
rangemax, qui disposent de fonctionnalités similaires à celles de la fonction
Max. Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
numérique ArgumentsArgumentDescriptionexprExpression ou champ contenant les
données à mesurer.rankLa valeur par défaut de rank est 1, qui correspond à la
valeur la plus élevée. Si vous spécifiez 2 comme valeur pour rank, la
deuxième valeur la plus élevée est renvoyée. Si la valeur de rank est égale à
3, on obtient la troisième valeur la plus élevée, et ainsi de
suite.SetExpressionPar défaut, la fonction d'agrégation couvre l'ensemble des
enregistrements possibles définis par la sélection. Il est possible de
définir un ensemble d'enregistrements alternatif à l'aide d'une expression
d'analyse d'ensembles. TOTALSi le terme TOTAL précède les arguments de la
fonction, le calcul est effectué à partir de toutes les valeurs possibles au
vu des sélections actives, et pas seulement à partir de celles qui sont
associées à la valeur dimensionnelle active. Autrement dit, les dimensions du
graphique ne sont pas prises en compte. En utilisant TOTAL [<fld {.fld}>], où
le qualificateur TOTAL est suivi d'un ou de plusieurs noms constituant un
sous-ensemble des variables de dimension du graphique, vous créez un sous-
ensemble du nombre total de valeurs
possibles. DonnéesCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Exemples et résultatsExemplesRésultatsMax
(UnitSales)10, car il s'agit de la valeur la plus élevée sous UnitSales.La
valeur d'une commande est calculée à partir du nombre d'unités vendues en
(UnitSales) multiplié par le prix unitaire.Max(UnitSales*UnitPrice)150, car
il s'agit de la valeur la plus élevée du résultat du calcul de toutes les
```

```
valeurs possibles pour (UnitSales)*(UnitPrice).Max(UnitSales, 2)9, qui
correspond à la deuxième valeur la plus élevée.Max(TOTAL UnitSales)10, car le
qualificateur TOTAL signifie que la valeur la plus élevée possible est
recherchée, sans tenir compte des dimensions du graphique. Pour un graphique
utilisant Customer comme dimension, le qualificateur TOTAL permet de garantir
le renvoi de la valeur maximale sur l'ensemble de données complet au lieu de
la valeur UnitSales maximale pour chaque client.Sélectionnez Customer B.Max
({1} TOTAL UnitSales)10, est le résultat, quelle que soit la sélection
effectuée, car l'expression Set Analysis {1} définit l'ensemble
d'enregistrements à évaluer comme ALL sans tenir compte de la
sélection.Données utilisées dans les exemples :ProductData:LOAD * inline
[Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|B
B|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility
|CC||19] (delimiter is '|'); FirstSortedValue RangeMax ([{SetExpression}]
[DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() permet de déterminer la valeur la plus basse parmi les données agrégées. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus basse.

```
Min - fonction de graphique([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr [,rank])
```

Mode

Mode() permet de déterminer la valeur la plus fréquente, la valeur de mode, contenue dans les données agrégées. La fonction **Mode()** peut aussi bien traiter des valeurs textuelles que des valeurs numériques.

```
Mode - fonction de graphique ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() renvoie une valeur s'il n'y a qu'un seul résultat possible dans les données agrégées. Par exemple, la recherche du seul produit dont le prix unitaire est égal à 9 renverra NULL si plusieurs produits ont un prix unitaire de 9.

```
Only - fonction de graphique([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

Sum

Sum() calcule le nombre total de valeurs fournies par l'expression ou le champ sur les données agrégées.

```
Sum - fonction de graphique([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr])
```

FirstSortedValue

FirstSortedValue() renvoie la valeur de l'expression spécifiée dans **value** qui correspond au résultat du tri de l'argument **sort_weight**, par exemple, le nom du produit ayant le prix unitaire le plus bas. Il est possible de spécifier la nième valeur dans l'ordre de tri dans l'argument **rank**. Si plusieurs valeurs résultantes

partagent le même champ **sort_weight** pour la fonction **rank** spécifiée, la fonction renvoie la valeur NULL. Les valeurs sont itérées sur un nombre d'enregistrements définis par une clause **group by** ou agrégées sur l'ensemble de données entier si aucune clause **group by** n'a été définie.

Syntaxe :

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
value Expression	La fonction détermine la valeur de l'expression **value** correspondant au résultat du tri du champ **sort_weight**.
sort-weight Expression	Expression contenant les données à trier. La première valeur (la plus faible) de **sort_weight** est identifiée, ce qui permet de déterminer la valeur correspondante de l'expression **value**. Si vous placez un signe moins devant **sort_weight**, la fonction renvoie alors la dernière valeur triée (la plus élevée).
rank Expression	Si vous spécifiez une valeur **rank** "n" supérieure à 1, vous obtenez la nième valeur triée.
distinct	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemples de script

Exemple	Résultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD|12|25|2 Canutility|AA|3|8|3 Canutility|CC|13|19|3 Divadip|AA|9|16|4 Divadip|AA|10|16|4 Divadip|DD|11|10|4] (delimiter is '|'); FirstSortedValue: LOAD Customer,FirstSortedValue(Product, UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithSmallestOrderByCustomer Astrida CC Betacab AA Canutility AA Divadip DD La fonction trie la colonne UnitSales de la plus petite à la plus grande valeur, recherchant la valeur de l'entrée Customer dotée de la plus petite valeur UnitSales, la plus petite commande. Car CC correspond à la plus petite commande (valeur de UnitSales=2) pour le client Astrida. AA correspond à la plus petite commande (4) du client Betacab, AA correspond à la plus petite commande (8) du client Canutility et DD correspond à la plus petite commande (10) du client Divadip..
Supposons que la table **Temp** est chargée comme dans l'exemple précédent : LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip - Comme un signe moins précède l'argument sort_weight, la fonction trie les valeurs les plus grandes en premier. Car AA correspond à la plus grande commande (valeur UnitSales égale à 18) du client Astrida, DD à la plus grande commande (12) du client Betacab et CC à la plus grande commande (13) du client Canutility. Il y a deux valeurs identiques pour la plus grande commande (16) du client Divadip, ce qui produit un résultat nul.

Exemple	Résultat

Supposons que la table **Temp** est chargée comme dans l'exemple précédent :
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	La situation est identique à celle de l'exemple précédent, sauf que le qualificateur distinct est utilisé. De ce fait, le doublon obtenu précédemment pour Divadip est ignoré, permettant le renvoi d'une valeur non nulle.

FirstSortedValue - fonction de graphique

FirstSortedValue() renvoie la valeur de l'expression spécifiée dans **value** qui correspond au résultat du tri de l'argument **sort_weight**, par exemple, le nom du produit ayant le prix unitaire le plus bas. Il est possible de spécifier la nième valeur dans l'ordre de tri dans l'argument **rank**. Si plusieurs valeurs résultantes partagent le même champ **sort_weight** pour la fonction **rank** spécifiée, la fonction renvoie la valeur NULL.

Syntaxe :

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
value	Champ de sortie. La fonction détermine la valeur de l'expression **value** correspondant au résultat du tri du champ **sort_weight**.
sort_weight	Champ de saisie. Expression contenant les données à trier. La première valeur (la plus faible) de **sort_weight** est identifiée, ce qui permet de déterminer la valeur correspondante de l'expression **value**. Si vous placez un signe moins devant **sort_weight**, la fonction renvoie alors la dernière valeur triée (la plus élevée).
rank	Si vous spécifiez une valeur **rank** "n" supérieure à 1, vous obtenez la nième valeur triée.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.

Argument	Description
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Données

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

Exemples et résultats

Exemple	Résultat
`firstsortedvalue (Product, UnitPrice)`	BB, qui correspond au produit `Product` doté du prix unitaire `UnitPrice` le plus bas (9).
`firstsortedvalue (Product, UnitPrice, 2)`	BB, qui correspond au produit `Product` doté du deuxième prix unitaire `UnitPrice` le plus bas (10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, qui correspond au client `Customer` disposant du produit `Product` doté du deuxième prix unitaire `UnitPrice` le plus élevé (20).
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, car il y a deux valeurs `Customer` (Astrida et Canutility) dotées du même rang rank (troisième prix unitaire) `UnitPrice` le plus bas (15). Le qualificateur distinct permet de garantir l'absence de résultats NULL inattendus.

Exemple	Résultat
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, qui correspond au client `Customer` disposant de la deuxième valeur de commande la plus élevée `UnitPrice` multipliée par `UnitSales` (120).

Données utilisées dans les exemples :

```
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

Max() permet de déterminer la valeur numérique la plus élevée contenue dans les données agrégées de l'expression définie par une clause **group by**. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus élevée.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
rank Expression	La valeur par défaut de **rank** est 1, qui correspond à la valeur la plus élevée. Si vous spécifiez 2 comme valeur pour **rank**, la deuxième valeur la plus élevée est renvoyée. Si la valeur de **rank** est égale à 3, on obtient la troisième valeur la plus élevée, et ainsi de suite.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemple :

```
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;
```

Table des résultats

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Exemple :

Supposons que la table **Temp** est chargée comme dans l'exemple précédent :

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Table des résultats

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - fonction de graphique

Max() permet de déterminer la valeur la plus élevée parmi les données agrégées. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus élevée.

*Il est également recommandé de consulter la description des fonctions **FirstSortedValue** et **rangemax**, qui disposent de fonctionnalités similaires à celles de la fonction **Max**.*

Syntaxe :

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
rank	La valeur par défaut de **rank** est 1, qui correspond à la valeur la plus élevée. Si vous spécifiez 2 comme valeur pour **rank**, la deuxième valeur la plus élevée est renvoyée. Si la valeur de **rank** est égale à 3, on obtient la troisième valeur la plus élevée, et ainsi de suite.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Données

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

Exemples et résultats

Exemples	Résultats
`Max(UnitSales)`	10, car il s'agit de la valeur la plus élevée sous `UnitSales`.
La valeur d'une commande est calculée à partir du nombre d'unités vendues en (`UnitSales`) multiplié par le prix unitaire. `Max (UnitSales*UnitPrice)`	150, car il s'agit de la valeur la plus élevée du résultat du calcul de toutes les valeurs possibles pour (`UnitSales`)*(`UnitPrice`).
`Max(UnitSales, 2)`	9, qui correspond à la deuxième valeur la plus élevée.
`Max(TOTAL UnitSales)`	10, car le qualificateur TOTAL signifie que la valeur la plus élevée possible est recherchée, sans tenir compte des dimensions du graphique. Pour un graphique utilisant Customer comme dimension, le qualificateur TOTAL permet de garantir le renvoi de la valeur maximale sur l'ensemble de données complet au lieu de la valeur UnitSales maximale pour chaque client.
Sélectionnez `Customer B`. `Max({1} TOTAL UnitSales)`	10, est le résultat, quelle que soit la sélection effectuée, car l'expression Set Analysis {1} définit l'ensemble d'enregistrements à évaluer comme ALL sans tenir compte de la sélection.

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *FirstSortedValue - fonction de graphique (page 333)*
p *RangeMax (page 1347)*

Min

Min() renvoie la valeur numérique la plus basse contenue dans les données agrégées de l'expression définie par une clause **group by**. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus basse.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
rank Expression	La valeur par défaut de **rank** est 1, qui correspond à la valeur la plus faible. Si vous spécifiez 2 comme valeur pour **rank**, la deuxième valeur la plus faible est renvoyée. Si la valeur de **rank** est égale à 3, on obtient la troisième valeur la plus faible, et ainsi de suite.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemple :

```
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;
```

Table des résultats

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Exemple :

Supposons que la table **Temp** est chargée comme dans l'exemple précédent :

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Table des résultats

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - fonction de graphique

Min() permet de déterminer la valeur la plus basse parmi les données agrégées. Si vous spécifiez un argument **rank** n, vous pouvez rechercher la énième valeur la plus basse.

*Il est également recommandé de consulter la description des fonctions **FirstSortedValue** et **rangemin**, qui disposent de fonctionnalités similaires à celles de la fonction **Min**.*

Syntaxe :

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
rank	La valeur par défaut de **rank** est 1, qui correspond à la valeur la plus faible. Si vous spécifiez 2 comme valeur pour **rank**, la deuxième valeur la plus faible est renvoyée. Si la valeur de **rank** est égale à 3, on obtient la troisième valeur la plus faible, et ainsi de suite.

Argument	Description
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Données

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

La fonction Min() doit renvoyer une valeur différente de NULL parmi le tableau de valeurs fourni par l'expression (le cas échéant). Ainsi, dans les exemples, comme les données comportent des valeurs NULL, la fonction renvoie la première valeur différente de NULL évaluée à partir de l'expression.

Exemples et résultats

Exemples	Résultats
`Min(UnitSales)`	2, car il s'agit de la valeur non NULL la plus basse sous `UnitSales`.

Exemples	Résultats
La valeur d'une commande est calculée à partir du nombre d'unités vendues en (`UnitSales`) multiplié par le prix unitaire. `Min (UnitSales*UnitPrice)`	40, car il s'agit du résultat non NULL le plus bas du calcul de toutes les valeurs possibles pour (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	4, car il s'agit de la deuxième valeur la plus basse (après les valeurs NULL).
`Min(TOTAL UnitSales)`	2, car le qualificateur TOTAL signifie que la valeur la plus basse possible est recherchée, sans tenir compte des dimensions du graphique. Pour un graphique utilisant Customer comme dimension, le qualificateur TOTAL permet de garantir le renvoi de la valeur minimale sur l'ensemble de données complet au lieu de la valeur UnitSales minimale pour chaque client.
Sélectionnez Customer B. `Min({1} TOTAL UnitSales)`	2, quelle que soit la sélection effectuée pour Customer B. L'expression Set Analysis {1} définit l'ensemble d'enregistrements à évaluer comme ALL sans tenir compte de la sélection.

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *FirstSortedValue - fonction de graphique (page 333)*
p *RangeMin (page 1350)*

Mode

Mode() renvoie la valeur la plus fréquente, la valeur de mode, contenue dans les données agrégées de l'expression définie par une clause **group by**. La fonction **Mode()** peut renvoyer aussi bien des valeurs numériques que des valeurs textuelles.

Syntaxe :

```
Mode ( expr)
```

Type de données renvoyé : double

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.

Limitations :

Si plusieurs valeurs présentent exactement la même fréquence, la valeur NULL est renvoyée.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemples de script

Exemple	Résultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD Canutility|DD|3|8 Canutility|CC] (delimiter is '|'); Mode: LOAD Customer, Mode(Product) as MyMostOftenSoldProduct Resident Temp Group By Customer;	MyMostOftenSoldProduct AA car AA est le seul produit vendu plusieurs fois.

Mode - fonction de graphique

Mode() permet de déterminer la valeur la plus fréquente, la valeur de mode, contenue dans les données agrégées. La fonction **Mode()** peut aussi bien traiter des valeurs textuelles que des valeurs numériques.

Syntaxe :

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

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Données

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

Exemples et résultats :

Exemples	Résultats
`Mode(UnitPrice)` Sélectionnez `Customer A`.	15, car il s'agit de la valeur la plus fréquente dans `UnitSales`. Renvoie NULL (-). Aucune valeur unique n'apparaît plus souvent que les autres.

`Mode(Product)`
Sélectionnez
`Customer A`
	AA, car il s'agit de la valeur la plus fréquente sous `Product`.
	
	Renvoie NULL (-). Aucune valeur unique n'apparaît plus souvent que les autres.

`Mode` `(TOTAL UnitPrice)`	15, car le qualificateur TOTAL signifie que la valeur la plus fréquente est toujours 15, même en ignorant les dimensions du graphique.

Sélectionnez
`Customer B`.
 	
`Mode({1}`
`TOTAL UnitPrice)`
	15, est le résultat, quelle que soit la sélection effectuée, car l'expression Set Analysis {1} définit l'ensemble d'enregistrements à évaluer comme ALL sans tenir compte de la sélection.

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *Avg - fonction de graphique (page 399)*
p *Median - fonction de graphique (page 439)*

Only

Only() renvoie une valeur s'il n'y a qu'un seul résultat possible dans les données agrégées. Si les enregistrements contiennent une seule valeur, c'est elle qui est renvoyée, sinon c'est la valeur NULL. Utilisez la clause **group by** si vous souhaitez que l'évaluation porte sur plusieurs enregistrements. La fonction **Only()** peut renvoyer des valeurs numériques comme des valeurs textuelles.

Syntaxe :

```
Only ( expr )
```

Type de données renvoyé : double

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

```
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;
```

Table des résultats

Customer	MyUniqIDCheck
Astrida	1 car seul le client Astrida dispose des enregistrements complets incluant le CustomerID.

Only - fonction de graphique

Only() renvoie une valeur s'il n'y a qu'un seul résultat possible dans les données agrégées. Par exemple, la recherche du seul produit dont le prix unitaire est égal à 9 renverra NULL si plusieurs produits ont un prix unitaire de 9.

Syntaxe :

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

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Utilisez Only() pour obtenir un résultat NULL lorsque plusieurs valeurs sont possibles dans les échantillons de données.

Exemples et résultats :

Données

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

Exemples et résultats

Exemples	Résultats
`Only ({<UnitPrice={9}>} Product)`	BB, car il s'agit du seul produit `Product` dont le prix unitaire `UnitPrice` est égal à '9'.
`Only({<Product={DD}>} Customer)`	Betacab, car il s'agit du seul client `Customer` vendant un produit `Product` appelé 'DD'.
`Only ({<UnitPrice={20}>} UnitSales)`	Le nombre d'éléments `UnitSales` pour lesquels `UnitPrice` est égal à 20 est de 2, car il n'y a qu'une seule valeur sous `UnitSales` pour laquelle le prix unitaire (`UnitPrice`) = 20.
`Only ({<UnitPrice={15}>} UnitSales)`	NULL, car il y a deux valeurs sous `UnitSales` pour lesquelles le prix unitaire (`UnitPrice`) = 15.

Données utilisées dans les exemples :

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Sum() calcule le nombre total de valeurs agrégées dans l'expression définie par une clause **group by**.

Syntaxe :

```
sum ( [ distinct] expr)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.
expr Expression	Expression ou champ contenant les données à mesurer.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

```
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;
```

Table des résultats

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - fonction de graphique

Sum() calcule le nombre total de valeurs fournies par l'expression ou le champ sur les données agrégées.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés. *Même si le qualificateur DISTINCT est pris en charge, utilisez-le avec une extrême prudence, car il peut induire le lecteur en erreur, en le laissant supposer qu'une valeur totale est affichée alors que certaines données ont été omises.*
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Données

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

Exemples et résultats

Exemples	Résultats
`Sum(UnitSales)`	38. Total des valeurs comprises dans `UnitSales`.

Exemples	Résultats
`Sum(UnitSales*UnitPrice)`	505. Total de la valeur `UnitPrice` multipliée par les valeurs `UnitSales` agrégées.
`Sum (TOTAL UnitSales*UnitPrice)`	505 pour toutes les lignes de la table de même que le total, car le qualificateur TOTAL implique que la somme est toujours égale à 505, quelles que soient les dimensions du graphique.
Sélectionnez `Customer B`. `Sum({1} TOTAL UnitSales*UnitPrice)`	505 est le résultat, quelle que soit la sélection effectuée, car l'expression Set Analysis {1} définit l'ensemble d'enregistrements à évaluer comme ALL sans tenir compte de la sélection.

Données utilisées dans les exemples :

```
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 '|');
```

Fonctions d'agrégation de décompte

Les fonctions d'agrégation de décompte renvoient différents types de décompte d'une expression portant sur plusieurs enregistrements dans un script de chargement de données ou un certain nombre de valeurs dans une dimension de graphique.

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Fonctions d'agrégation de décompte utilisées dans le script de chargement de données

Count

Count() renvoie le nombre de valeurs agrégées dans l'expression définie par une clause **group by**.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() renvoie le nombre de valeurs manquantes agrégées dans l'expression définie par une clause **group by**.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() renvoie le nombre de valeurs NULL agrégées dans l'expression définie par une clause **group by**.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() renvoie le nombre de valeurs numériques identifiées dans l'expression définie par une clause **group by**.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() renvoie le nombre de valeurs de champ non numériques agrégées dans l'expression définie par une clause **group by**.

```
TextCount ([ distinct ] expression)
```

Fonctions d'agrégation de décompte utilisées dans les expressions de graphique

Les fonctions d'agrégation de décompte suivantes peuvent s'utiliser dans les graphiques :

Count

Count() permet d'agréger le nombre de valeurs, textuelles et numériques, dans chaque dimension du graphique.

```
Count - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

MissingCount

MissingCount() permet d'agréger le nombre de valeurs manquantes dans chaque dimension du graphique. Les valeurs manquantes sont toutes des valeurs non numériques.

```
MissingCount - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() permet d'agréger le nombre de valeurs NULL dans chaque dimension du graphique.

```
NullCount - fonction de graphique({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() permet d'agréger le nombre de valeurs numériques dans chaque dimension du graphique.

```
NumericCount - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() permet d'agréger le nombre de valeurs de champ non numériques dans chaque dimension du graphique.

```
TextCount - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Count

Count() renvoie le nombre de valeurs agrégées dans l'expression définie par une clause **group by**.

Syntaxe :

```
Count( [distinct ] expr)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemples de script

Exemple	Résultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB|1|25| 25 Canutility|AA|3|8|15 Canutility|CC|||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); Count1: LOAD Customer,Count(OrderNumber) as OrdersByCustomer Resident Temp Group By Customer;	Customer OrdersByCustomer Astrida 3 Betacab 3 Canutility 2 Divadip 2 Du moment que la dimension Customer est incluse dans la table sur la feuille, sinon le résultat pour OrdersByCustomer correspond à 3, 2.
Supposons que la table **Temp** est chargée comme dans l'exemple précédent : LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
Supposons que la table **Temp** est chargée comme dans le premier exemple : LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 Parce qu'il existe deux valeurs de OrderNumber avec la même valeur, 1, et une valeur nulle.

Count - fonction de graphique

Count() permet d'agréger le nombre de valeurs, textuelles et numériques, dans chaque dimension du graphique.

Syntaxe :

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

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD	1	25	25
Canutility	AA	3	8	15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Dans les exemples suivants, nous supposons que tous les clients sont sélectionnés, sauf mention contraire.

Exemples et résultats

Exemple	Résultat
`Count(OrderNumber)`	10, car il y a 10 champs qui pourraient disposer d'une valeur pour OrderNumber, tous les enregistrements, même ceux qui sont vides, sont comptabilisés. *"0" est interprété comme une valeur et non comme une cellule vide. Cependant, si une mesure est agrégée sur la valeur 0 pour une dimension, cette dernière ne sera pas incluse dans les graphiques.*
`Count(Customer)`	10, car la fonction Count évalue le nombre d'occurrences dans tous les champs.
`Count(DISTINCT [Customer])`	4, car avec l'utilisation du qualificateur Distinct, Count prend seulement en compte les occurrences uniques.
Supposons que le client Canutility est sélectionné. `Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	0.2, car l'expression renvoie le nombre de commandes concernant le client sélectionné sous forme de pourcentage par rapport aux commandes de l'ensemble des clients. Dans ce cas, le résultat est 2/10.
Supposons que les clients Astrida et Canutility sont sélectionnés. `Count(TOTAL <Product> OrderNumber)`	5, car il s'agit du nombre de commandes passées pour les produits des seuls clients sélectionnés, les cellules vides étant comptabilisées.

Données utilisées dans les exemples :

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

MissingCount

MissingCount() renvoie le nombre de valeurs manquantes agrégées dans l'expression définie par une clause **group by**.

Syntaxe :

```
MissingCount ( [ distinct ] expr)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemples de script

Exemple	Résultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB||| 25 Canutility|AA|||15 Canutility|CC| ||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); MissCount1: LOAD Customer,MissingCount(OrderNumber) as MissingOrdersByCustomer Resident Temp Group By Customer; Load MissingCount(OrderNumber) as TotalMissingCount Resident Temp;	Customer MissingOrdersByCustomer Astrida 0 Betacab 1 Canutility 2 Divadip 0 La seconde instruction produit le résultat suivant : TotalMissingCount 3 dans une table comportant cette dimension.

Exemple	Résultat

Supposons que la table **Temp** est chargée comme dans l'exemple précédent :
`LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;`
	`TotalMissingCountDistinct`
	`1`
	Car il n' y a qu'une seule valeur OrderNumber manquante.

MissingCount - fonction de graphique

MissingCount() permet d'agréger le nombre de valeurs manquantes dans chaque dimension du graphique. Les valeurs manquantes sont toutes des valeurs non numériques.

Syntaxe :

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

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Data

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

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

Exemples et résultats

Exemple	Résultat
`MissingCount([OrderNumber])`	3, car 3 des 10 champs OrderNumber sont vides. *"0" est interprété comme une valeur et non comme une cellule vide. Cependant, si une mesure est agrégée sur la valeur 0 pour une dimension, cette dernière ne sera pas incluse dans les graphiques.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	L'expression renvoie le nombre de commandes incomplètes concernant le client sélectionné sous forme de fraction par rapport aux commandes incomplètes de l'ensemble des clients. Il manque au total 3 valeurs sous OrderNumber pour l'ensemble des clients. Ainsi, pour chaque client (Customer) doté d'une valeur manquante sous Product, le résultat est égal à 1/3.

Données utilisées dans l'exemple :

```
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

NullCount() renvoie le nombre de valeurs NULL agrégées dans l'expression définie par une clause **group by**.

Syntaxe :

```
NullCount ( [ distinct ] expr)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemples de script

Exemple	Résultat

`Set NULLINTERPRET = NULL;`
`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD|||`
`Canutility|AA|3|8|`
`Canutility|CC|NULL||`
`] (delimiter is '|');`
`Set NULLINTERPRET=;`
`NullCount1:`
`LOAD Customer,NullCount(OrderNumber) as NullOrdersByCustomer Resident Temp Group By Customer;`
 	
`LOAD NullCount(OrderNumber) as TotalNullCount Resident Temp;`
	`Customer NullOrdersByCustomer`
	`Astrida 0`
	`Betacab 0`
	`Canutility 1`
	
	La seconde instruction produit le résultat suivant :
	`TotalNullCount`
	`1`
	dans une table comportant cette dimension, car un seul enregistrement contient une valeur nulle.

NullCount - fonction de graphique

NullCount() permet d'agréger le nombre de valeurs NULL dans chaque dimension du graphique.

Syntaxe :

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

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
set_ expression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Exemples et résultats

Exemple	Résultat
NullCount ([OrderNumber])	1, car nous avons introduit une valeur « null » à l'aide de NullInterpret dans l'instruction **LOAD** inline.

Données utilisées dans l'exemple :

```
Set NULLINTERPRET = NULL;
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
```

```
Betacab|DD|||
Canutility|AA|3|8|
Canutility|CC|NULL||
] (delimiter is '|');
Set NULLINTERPRET=;
```

NumericCount

NumericCount() renvoie le nombre de valeurs numériques identifiées dans l'expression définie par une clause **group by**.

Syntaxe :

```
NumericCount ( [ distinct ] expr)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemple de script

Exemple	Résultat
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	La seconde instruction produit le résultat suivant : TotalNumericCount 7 dans une table comportant cette dimension.
Supposons que la table **Temp** est chargée comme dans l'exemple précédent : `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Comme un OrderNumber en duplique un autre, le résultat de 6 signifie qu'il n'y a pas de doublon.

Exemple :

```
Temp:
LOAD * inline [
```

```
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|7|1|25
] (delimiter is '|');
NumCount1:
LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Table des résultats

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - fonction de graphique

NumericCount() permet d'agréger le nombre de valeurs numériques dans chaque dimension du graphique.

Syntaxe :

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

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
set_ expression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.

Argument	Description
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Dans les exemples suivants, nous supposons que tous les clients sont sélectionnés, sauf mention contraire.

Exemples et résultats

Exemple	Résultat
`NumericCount ([OrderNumber])`	7, car trois des 10 champs figurant sous OrderNumber sont vides. *"0" est interprété comme une valeur et non comme une cellule vide. Cependant, si une mesure est agrégée sur la valeur 0 pour une dimension, cette dernière ne sera pas incluse dans les graphiques.*

Exemple	Résultat
`NumericCount ([Product])`	0, car tous les noms de produit sont de type texte. En général, cette fonction est employée pour vérifier qu'aucun champ de texte ne contient de valeur numérique.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Compte le nombre de numéros de commande numériques distincts et le divise par le nombre de numéros de commande numériques et non numériques. Le résultat sera égal à 1 si toutes les valeurs de champ sont numériques. En général, cette fonction est employée pour vérifier que toutes les valeurs de champ sont numériques. Dans cet exemple, il y a 7 valeurs numériques distinctes pour OrderNumber sur 8 valeurs numériques distinctes et sans ID numérique, donc l'expression renvoie 0.875.

Données utilisées dans l'exemple :

```
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

TextCount() renvoie le nombre de valeurs de champ non numériques agrégées dans l'expression définie par une clause **group by**.

Syntaxe :

```
TextCount ( [ distinct ] expr)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr Expression	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemple :

```
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;
```

Table des résultats

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Exemple :

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Table des résultats

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - fonction de graphique

TextCount() permet d'agréger le nombre de valeurs de champ non numériques dans chaque dimension du graphique.

Syntaxe :

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

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25

Customer	Product	OrderNumber	UnitSales	Unit Price
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Exemples et résultats

Exemple	Résultat
`TextCount ([Product])`	10, car les 10 champs figurant sous Product sont tous de type texte. *"0" est interprété comme une valeur et non comme une cellule vide. Cependant, si une mesure est agrégée sur la valeur 0 pour une dimension, cette dernière ne sera pas incluse dans les graphiques. Les cellules vides sont évaluées comme non textuelles et ne sont pas comptabilisées par la fonction TextCount.*
`TextCount ([OrderNumber])`	3, car les cellules vides sont comptabilisées. En général, cette fonction est employée pour vérifier qu'aucun champ numérique ne contient de valeurs de texte ou n'est différent de zéro.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Compte le nombre de valeurs de texte distinctes de produit sous Product (4) et le divise par le nombre total de valeurs figurant sous Product (10). Le résultat est égal à 0.4.

Données utilisées dans l'exemple :

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|1|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|||| 25
Canutility|AA|||15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Fonctions d'agrégation financières

Cette section décrit les fonctions d'agrégation des opérations financières concernant les paiements et les flux de liquidités.

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Fonctions d'agrégation financières utilisées dans le script de chargement de données

IRR

La fonction **IRR()** renvoie le taux de rendement interne agrégé pour une série de flux de liquidités que représentent les valeurs de l'expression itérée sur un nombre donné d'enregistrements définis par une clause group by.

```
IRR (expression)
```

XIRR

La fonction **XIRR()** renvoie le taux de rendement interne agrégé pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions **pmt** et **date** itérées sur un nombre donné d'enregistrements définis par une clause group by. Tous les paiements sont actualisés sur une base de 365 jours par an.

```
XIRR (valueexpression, dateexpression )
```

NPV

La fonction de script **NPV()** prend une taux de remise et plusieurs valeurs triées par période. Les entrées (revenus) sont positives et les sorties (paiements futurs) sont supposées être des valeurs négatives pour ces calculs. Ils se produisent à la fin de chaque période.

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

XNPV

La fonction de script **XNPV()** prend des dates spécifiques correspondant à chaque flux de trésorerie faisant l'objet d'une remise en dehors du taux de remise. Elle est différente de la fonction **NPV()**, car la fonction **NPV()** suppose que toutes les périodes sont égales. C'est pourquoi la fonction **XNPV()** est plus précise que la fonction **NPV()**.

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

Fonctions d'agrégation financières utilisées dans les expressions de graphique

Ces fonctions d'agrégation financières peuvent s'utiliser dans les graphiques.

IRR

IRR() renvoie le taux de rendement interne agrégé pour une série de flux de liquidités que représentent les nombres de l'expression fournie par l'argument **value** itéré sur les dimensions du graphique.

```
IRR - fonction de graphique([TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() renvoie la valeur actuelle nette agrégée d'un investissement basée sur un argument **discount_rate** par période et une série de paiements (valeurs négatives) et de revenus (valeurs positives) ultérieurs que représentent les nombres figurant dans l'argument **value**, itéré sur les dimensions du graphique. Les paiements et les revenus sont censés intervenir à la fin de chaque période.

```
NPV - fonction de graphique([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() renvoie le taux de rendement interne agrégé pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions fournies par **pmt** et **date**, itérées sur les dimensions du graphique. Tous les paiements sont actualisés sur une base de 365 jours par an.

```
XIRR - fonction de graphique (page 382)([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() renvoie la valeur actuelle nette agrégée pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions fournies par **pmt** et **date**, itérées sur les dimensions du graphique. Tous les paiements sont actualisés sur une base de 365 jours par an.

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

IRR

La fonction **IRR()** renvoie le taux de rendement interne agrégé pour une série de flux de liquidités que représentent les valeurs de l'expression itérée sur un nombre donné d'enregistrements définis par une clause group by.

Ces flux de liquidités ne doivent pas nécessairement être égaux, comme ils le seraient pour une annuité. Cependant, les flux de liquidités doivent intervenir à intervalle régulier, mensuellement ou annuellement, par exemple. Le taux de rendement interne correspond au taux d'intérêt perçu pour un investissement consistant en des paiements (valeurs négatives) et des revenus (valeurs positives) qui interviennent à intervalle régulier. La fonction nécessite au moins une valeur positive et une valeur négative à calculer.

Syntaxe :

```
IRR(value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Expression ou champ contenant les données à mesurer.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Exemples et résultats :

Exemples et résultats

Exemple	Année	IRR2013

`Cashflow:`
`LOAD 2013 as Year, * inline [`
`Date|Discount|Payments`
`2013-01-01|0.1|-10000`
`2013-03-01|0.1|3000`
`2013-10-30|0.1|4200`
`2014-02-01|0.2|6800`
`] (delimiter is '|');`
 	
`Cashflow1:`
`LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;`
	2013	0.1634

IRR - fonction de graphique

IRR() renvoie le taux de rendement interne agrégé pour une série de flux de liquidités que représentent les nombres de l'expression fournie par l'argument **value** itéré sur les dimensions du graphique.

Ces flux de liquidités ne doivent pas nécessairement être égaux, comme ils le seraient pour une annuité. Cependant, les flux de liquidités doivent intervenir à intervalle régulier, mensuellement ou annuellement, par exemple. Le taux de rendement interne correspond au taux d'intérêt perçu pour un investissement consistant en des paiements (valeurs négatives) et des revenus (valeurs positives) qui interviennent à intervalle régulier. La fonction nécessite au moins une valeur positive et une valeur négative à calculer.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Expression ou champ contenant les données à mesurer.

Argument	Description
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples et résultats :

Exemples et résultats

Exemple	Résultat
`IRR (Payments)`	0.1634 Par nature, les paiements sont supposés être périodiques, mensuels par exemple. *Le champ Date est utilisé dans l'exemple XIRR où les paiements peuvent être non périodiques du moment que vous indiquez les dates auxquelles ils doivent être effectués.*

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *XIRR - fonction de graphique (page 382)*
p *Aggr - fonction de graphique (page 545)*

NPV

La fonction de script **NPV()** prend une taux de remise et plusieurs valeurs triées par période. Les entrées (revenus) sont positives et les sorties (paiements futurs) sont supposées être des valeurs négatives pour ces calculs. Ils se produisent à la fin de chaque période.

La valeur Net Present Value, or NPV, est utilisée pour calculer la valeur totale actuelle de flux de trésorerie futurs. Pour calculer la valeur NPV, nous devons estimer les flux de trésorerie futurs pour chaque période et déterminer le taux de remise correct. La fonction de script **NPV()** prend une taux de remise et plusieurs valeurs triées par période. Les entrées (revenus) sont positives et les sorties (paiements futurs) sont supposées être des valeurs négatives pour ces calculs. Ces derniers se produisent à la fin de chaque période.

Syntaxe :

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

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise.

La formule permettant de calculer la valeur NPV est la suivante :

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

où :

- R_t = entrées/sorties de flux de trésorerie nettes au cours d'une seule période t
- i = taux de remise ou retour susceptible d'être obtenu dans des investissements alternatifs
- t = nombre de périodes

Arguments

Argument	Description
discount_rate	**discount_rate** est le taux de pourcentage de remise appliqué. Une valeur égale à 0.1 indique un taux de remise de 10 %.
value	Ce champ contient les valeurs de plusieurs périodes triées par période. La première valeur est supposée être le flux de trésorerie à la fin de la période 1, etc.

Limitations :

La fonction `NPV()` présente les restrictions suivantes :

- Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.
- Les valeurs de flux de trésorerie doivent être triées dans l'ordre de période croissant.

Cas d'utilisation

NPV() est une fonction financière utilisée pour vérifier la rentabilité des projets et dériver d'autres mesures. Cette fonction est utile en cas de disponibilité de flux de trésorerie sous forme de données brutes.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction SET DateFormat de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – un seul paiement (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'un projet et de son flux de trésorerie pour une période, chargé dans une table appelée CashFlow.
- Chargement résident de la table CashFlow, utilisé pour calculer le champ NPV du projet dans une table appelée NPV.
- Taux de remise codé en dur de 10 %, utilisé dans le calcul NPV.
- Instruction Group By, utilisée pour regrouper l'ensemble des paiements du projet.

Script de chargement

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

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- PrjId
- NPV

Tableau de résultats

PrjId	NPV
1	$909.09

Pour un seul paiement de 1 000 $ à recevoir à la fin d'une période, à un taux de remise de 10 % par période, la valeur NPV est égale à 1 000 $ divisé par (1 + taux de remise). La valeur NPV effective est égale à 909,09 $.

Exemple 2 – plusieurs paiements (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'un projet et de son flux de trésorerie pour plusieurs périodes, chargé dans une table appelée CashFlow.
- Chargement résident de la table CashFlow, utilisé pour calculer le champ NPV du projet dans une table appelée NPV.
- Un taux de remise codé en dur de 10 % (0,1) est utilisé dans le calcul NPV.
- Instruction Group By, utilisée pour regrouper l'ensemble des paiements du projet.

Script de chargement

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

```
1,2,1000
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- PrjId
- NPV

Tableau de résultats

PrjId	NPV
1	$1735.54

Pour des paiements de 1 000 $ à recevoir à la fin de deux périodes, à un taux de remise de 10 % par période, la valeur NPV effective est égale à 1 735,54 $.

Exemple 3 – plusieurs paiements (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Taux de remise pour deux projets, chargés dans une table appelée `Project`.
- Flux de trésorerie pour plusieurs périodes pour chaque projet par ID de projet et ID de période. Cet ID de période peut être utilisé pour trier les enregistrements si les données ne sont pas triées.
- Combinaison de `NoConcatenate`, de chargements résidents et de la fonction `Left Join` pour créer une table temporaire, `tmpNPV`. La table combine les enregistrements des tables `Project` et `CashFlow` en une seule table uniforme. Les taux de remise de cette table sont répétés pour chaque période.
- Chargement résident de la table `tmpNPV`, utilisé pour calculer le champ NPV de chaque projet dans une table appelée `NPV`.
- Taux de remise de valeur unique associé à chaque projet. Cette valeur est récupérée via la fonction `only()` et utilisée dans le calcul NPV pour chaque projet.
- Instruction `Group By`, utilisée pour regrouper l'ensemble des paiements pour chaque projet par ID de projet.

Pour éviter le chargement de toute donnée synthétique ou redondante dans le modèle de données, la table `tmpNPV` est abandonnée à la fin du script.

Script de chargement

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

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
1,3,1000
2,1,500
2,2,500
2,3,1000
2,4,1000
];

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

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

Drop table tmpNPV;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `PrjId`
- `NPV`

Tableau de résultats

PrjId	NPV
1	$2486.85
2	$2042.12

L'ID de projet 1 prévoit la réception de paiements de 1 000 $ à la fin de trois périodes, à un taux de remise de 10 % par période. Par conséquent, la valeur NPV effective est égale à 2 486,85 $.

L'ID de projet 2 prévoit deux paiements de 500 $ et deux autres paiements de 1 000 $ sur quatre périodes à un taux de remise de 15 %. Par conséquent, la valeur NPV effective est égale à 2 042,12 $.

Exemple 4 – Exemple rentabilité de projet (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Taux de remise et investissements initiaux (période 0) pour deux projets, chargés dans une table appelée `Project`.
- Flux de trésorerie pour plusieurs périodes pour chaque projet par ID de projet et ID de période. Cet ID de période peut être utilisé pour trier les enregistrements si les données ne sont pas triées.
- Combinaison de `NoConcatenate`, de chargements résidents et de la fonction `Left Join` pour créer une table temporaire, `tmpNPV`. La table combine les enregistrements des tables `Project` et `CashFlow` en une seule table uniforme. Les taux de remise de cette table sont répétés pour chaque période.
- Taux de remise de valeur unique associé à chaque projet, récupéré via la fonction `only()` et utilisé dans le calcul NPV pour chaque projet.
- Un chargement résident de la table tmpNPV est utilisé pour calculer le champ NPV de chaque projet dans une table appelée NPV.
- Un champ supplémentaire, qui divise la valeur NPV par l'investissement initial de chaque projet, est créé pour calculer l'indice de rentabilité du projet.
- Une instruction group by, effectuant un regroupement par ID de projet, est utilisée pour regrouper tous les paiements pour chaque projet.

Pour éviter le chargement de toute donnée synthétique ou redondante dans le modèle de données, la table `tmpNPV` est abandonnée à la fin du script.

Script de chargement

```
Project:
Load * inline [
PrjId,Discount_Rate, Initial_Investment
1,0.1,100000
```

```
2,0.15,100000
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values,
1,1,35000
1,2,35000
1,3,35000
2,1,30000
2,2,40000
2,3,50000
2,4,60000
];

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

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

Drop table tmpNPV;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- PrjId
- NPV

Créez la mesure suivante :

```
=only(Profitability_Index)
```

Tableau de résultats

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

L'ID de projet 1 a une valeur NPV effective de 87 039,82 $ et un investissement initial de 100 000 $. Par conséquent, l'index de rentabilité est égal à 0,87. Étant donné qu'il est inférieur à 1, le projet n'est pas rentable.

L'ID de projet 2 a une valeur NPV effective de 123 513,71 $ et un investissement initial de 100 000 $. Par conséquent, l'index de rentabilité est égal à 1,24. Étant donné qu'il est supérieur à 1, le projet est rentable.

NPV - fonction de graphique

NPV() renvoie la valeur actuelle nette agrégée d'un investissement basée sur un argument **discount_rate** par période et une série de paiements (valeurs négatives) et de revenus (valeurs positives) ultérieurs que représentent les nombres figurant dans l'argument **value**, itéré sur les dimensions du graphique. Les paiements et les revenus sont censés intervenir à la fin de chaque période.

Syntaxe :

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

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise.

Arguments :

Arguments

Argument	Description
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** est le taux de pourcentage de remise appliqué.
value	Expression ou champ contenant les données à mesurer.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles. Le qualificateur **TOTAL** peut être suivi d'une liste d'un ou de plusieurs noms de champ placés entre crochets angulaires. Ces noms de champ doivent constituer un sous-ensemble des variables de dimension du graphique. Dans ce cas, toutes les variables de dimension du graphique sont ignorées lors du calcul, à l'exception de celles figurant dans la liste. Autrement dit, une valeur est renvoyée pour chaque combinaison de valeurs de champ dans les champs de dimension de la liste. Il est par ailleurs possible d'inclure dans la liste des champs qui ne constituent pas une dimension dans un graphique. Cette option peut s'avérer utile dans le cas de dimensions groupées, où les champs de dimension ne sont pas fixes. Si vous listez toutes les variables du groupe, la fonction se déclenche lors de tout changement de niveau hiérarchique.

Limitations :

discount_rate et **value** ne doivent pas comprendre de fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples et résultats :

Exemples et résultats

Exemple	Résultat
`NPV(Discount, Payments)`	-$540.12

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *XNPV - fonction de graphique (page 390)*
p *Aggr - fonction de graphique (page 545)*

XIRR

La fonction **XIRR()** renvoie le taux de rendement interne agrégé pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions **pmt** et **date** itérées sur un nombre donné d'enregistrements définis par une clause group by. Tous les paiements sont actualisés sur une base de 365 jours par an.

Syntaxe :

```
XIRR(pmt, date )
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
pmt	Payments. Expression ou champ contenant le flux de liquidités correspondant au calendrier de dates de paiement spécifié dans la fonction **date**.
date	Expression ou champ contenant le calendrier de dates correspondant aux paiements de flux de liquidités spécifiés dans la fonction **pmt**.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes d'un côté ou des deux côtés d'une paire de données ont pour effet d'écarter la paire de données entière.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Exemples et résultats

Exemple	Année	XIRR2013

`Cashflow:`
`LOAD 2013 as Year, * inline [`
`Date|Discount|Payments`
`2013-01-01|0.1|-10000`
`2013-03-01|0.1|3000`
`2013-10-30|0.1|4200`
`2014-02-01|0.2|6800`
`] (delimiter is '|');`
 	
`Cashflow1:`
`LOAD Year,XIRR(Payments, Date) as XIRR2013 Resident Cashflow Group By Year;`
	2013	0.5385

XIRR - fonction de graphique

XIRR() renvoie le taux de rendement interne agrégé pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions fournies par **pmt** et **date**, itérées sur les dimensions du graphique. Tous les paiements sont actualisés sur une base de 365 jours par an.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
pmt	Payments. Expression ou champ contenant le flux de liquidités correspondant au calendrier de dates de paiement spécifié dans la fonction **date**.
date	Expression ou champ contenant le calendrier de dates correspondant aux paiements de flux de liquidités spécifiés dans la fonction **pmt**.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

pmt et **date** ne doivent pas comprendre de fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Exemples et résultats

Exemple	Résultat
`XIRR(Payments, Date)`	0.5385

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *IRR - fonction de graphique (page 371)*
p *Aggr - fonction de graphique (page 545)*

XNPV

La fonction de script **XNPV()** prend des dates spécifiques correspondant à chaque flux de trésorerie faisant l'objet d'une remise en dehors du taux de remise. Elle est différente de la fonction **NPV()**, car la fonction **NPV()** suppose que toutes les périodes sont égales. C'est pourquoi la fonction **XNPV()** est plus précise que la fonction **NPV()**.

Syntaxe :

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

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise.

La formule pour calculer XNPV est la suivante :

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

où :

- P_i = entrées/sorties de flux de trésorerie nettes au cours d'une seule période `i`
- d_1 = première date de paiement
- d_i = `i`e date de paiement
- `rate` = taux de remise

La valeur Net Present Value, or NPV, est utilisée pour calculer la valeur totale actuelle de flux de trésorerie futurs. Pour calculer la valeur NPV, nous devons estimer les flux de trésorerie futurs pour chaque période et déterminer le taux de remise correct.

`XNPV()` prend une taux de remise et plusieurs valeurs triées par période. Les entrées (revenus) sont positives et les sorties (paiements futurs) sont supposées être des valeurs négatives. Ces derniers se produisent à la fin de chaque période.

Arguments

Argument	Description
discount_rate	**discount_rate** est le taux de pourcentage de remise appliqué. Une valeur égale à 0.1 indique un taux de remise de 10 %.

Argument	Description
value	Ce champ contient les valeurs de flux de trésorerie. La première valeur est supposée être le flux de trésorerie de départ, et la date correspondante est utilisée comme référence pour calculer la valeur présente pour tous les flux de trésorerie futurs. *XNPV() n'applique pas de remise au flux de trésorerie initial. Les paiements ultérieurs font l'objet d'une remise en fonction d'une année de 365 jours. Cela est différent de NPV(), où chaque paiement fait l'objet d'une remise.*
date	Ce champ contient la date du flux de trésorerie (**value**, le deuxième paramètre). La première valeur est utilisée comme date de début pour calculer les décalages des flux de trésorerie futurs.

Limitations :

En présence de valeurs de texte et de valeurs NULL et en l'absence de valeurs pour une part et/ou l'autre part d'une paire de données, la paire de données toute entière sera ignorée.

Cas d'utilisation

- La valeur `XNPV()` est utilisée dans la modélisation financière pour calculer la valeur Net Present Value (NPV - Valeur actuelle nette) d'une opportunité d'investissement.
- En raison de sa plus grande précision, la valeur XNPV est préférée à la valeur NPV pour tous les types de modèles financiers.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – un seul paiement (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'un projet et son flux de trésorerie pour un an, dans une table appelée `CashFlow`. La date initiale du calcul est définie sur le 1er juillet 2022, avec un flux de trésorerie net égal à 0. Au bout d'un an, on a un flux de trésorerie de 1 000 $.
- Chargement résident de la table `CashFlow`, utilisé pour calculer le champ `XNPV` du projet dans une table appelée `XNPV`.
- Taux de remise codé en dur de 10 % (0,1), utilisé dans le calcul XNPV.
- Une instruction `Group By` est utilisée pour regrouper l'ensemble des paiements du projet.

Script de chargement

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2023',1000
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `PrjId`
- `XNPV`

Tableau de résultats

PrjId	XNPV
1	$909.09

Conformément à la formule, la valeur XNPV du premier enregistrement est 0, et, pour le deuxième enregistrement, la valeur XNPV est de 909,09 $. Ainsi, la valeur XNPV totale est de 909,09 $.

Exemple 2 – plusieurs paiements (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'un projet et son flux de trésorerie pour un an, dans une table appelée `CashFlow`.
- Chargement résident de la table `CashFlow`, utilisé pour calculer le champ `XNPV` du projet dans une table appelée `XNPV`.
- Taux de remise codé en dur de 10 % (0,1), utilisé dans le calcul XNPV.
- Une instruction `Group By` est utilisée pour regrouper l'ensemble des paiements du projet.

Script de chargement

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2024',500
1,'07/01/2023',1000
];

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

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `PrjId`
- `XNPV`

Tableau de résultats

PrjId	XNPV
1	$1322.21

Dans cet exemple, un paiement de 1 000 $ est reçu à la fin de la première année et un paiement de 500 $ à la fin de la deuxième année. Avec un taux de remise de 10 % par période, la valeur XNPV effective est égale à 1 322,21 $.

Notez que seule la première ligne de données doit faire référence à la date de base pour les calculs. Pour le reste des lignes, l'ordre n'a pas d'importance, car le paramètre date est utilisé pour calculer la période écoulée.

Exemple 3 – plusieurs paiements et flux de trésorerie irréguliers (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Taux de remise pour deux projets dans une table appelée `Project`.
- Flux de trésorerie pour plusieurs périodes pour chaque projet par ID de projet et dates. Le champ `Dates` est utilisé pour calculer la durée pendant laquelle le taux de remise est appliqué au flux de trésorerie. À l'exception du premier enregistrement (flux de trésorerie initial et date initiale), l'ordre des enregistrements n'a pas d'importance et sa modification ne devrait pas avoir d'impact sur les calculs.
- Via une combinaison de `NoConcatenate`, de chargements résidents et de la fonction `Left Join`, une table temporaire, `tmpNPV`, est créée, combinant les enregistrements des tables `Project` et `CashFlow` en une seule table uniforme. Les taux de remise de cette table sont répétés pour chaque flux de trésorerie.
- Chargement résident de la table `tmpNPV`, utilisé pour calculer le champ `XNPV` de chaque projet dans une table appelée `XNPV`.
- Le taux de remise de valeur unique associé à chaque projet est récupéré via la fonction `only()` et utilisé dans le calcul `XNPV` pour chaque projet.
- Une instruction `Group By`, effectuant un regroupement par ID de projet, est utilisée pour regrouper tous les paiements et dates correspondantes pour chaque projet.
- Pour éviter le chargement de toute donnée synthétique ou redondante dans le modèle de données, la table `tmpXNPV` est abandonnée à la fin du script.

Script de chargement

```
Project:
Load * inline [
PrjId,Discount_Rate
```

```
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2021',0
1,'07/01/2022',1000
1,'07/01/2023',1000
2,'07/01/2020',0
2,'07/01/2023',500
2,'07/01/2024',1000
2,'07/01/2022',500
];

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

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

Drop table tmpXNPV;
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- PrjId
- XNPV

Tableau de résultats

PrjId	XNPV
1	$1735.54
2	$278.36

L'ID de projet 1 a un flux de trésorerie initial de 0 $ le 1er juillet 2021. Deux paiements de 1 000 $ doivent être reçus à la fin de deux années consécutives, à un taux de remise de 10 % par période. Par conséquent, la valeur XNPV effective est égale à 1 735,54 $.

L'ID de projet 2 a un flux sortant initial de 1 000 $ (d'où le signe négatif) le 1er juillet 2020. Au bout de deux ans, un paiement de 500 $ est prévu. Au bout de trois ans, un autre paiement de 500 $ est prévu. Pour finir, le 1er juillet 2024, un paiement de 1 000 $ est prévu. Avec le taux de remise de 15 %, la valeur XNPV effective est égale à 278,36 $.

XNPV - fonction de graphique

XNPV() renvoie la valeur actuelle nette agrégée pour un calendrier de flux de liquidités (non nécessairement périodique) que représentent des nombres appariés dans les expressions fournies par **pmt** et **date**, itérées sur les dimensions du graphique. Tous les paiements sont actualisés sur une base de 365 jours par an.

Syntaxe :

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

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise.

Arguments :

Arguments

Argument	Description
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** est le taux de pourcentage de remise appliqué.
pmt	Payments. Expression ou champ contenant le flux de liquidités correspondant au calendrier de dates de paiement spécifié dans la fonction **date**.
date	Expression ou champ contenant le calendrier de dates correspondant aux paiements de flux de liquidités spécifiés dans la fonction **pmt**.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

discount_rate, **pmt** et **date** ne doivent pas comprendre de fonctions d'agrégation, à moins que ces agrégations internes ne contiennent les qualificateurs **TOTAL** ou **ALL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Exemples et résultats

Exemple	Résultat
XNPV(Discount, Payments, Date)	-$3164.35

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *NPV - fonction de graphique (page 380)*
p *Aggr - fonction de graphique (page 545)*

Fonctions d'agrégation statistiques

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Fonctions d'agrégation statistiques utilisées dans le script de chargement de données

Les fonctions d'agrégation statistiques suivantes peuvent s'utiliser dans les scripts.

Avg

Avg() permet de déterminer la valeur moyenne des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
Avg  ([distinct] expression)
```

Correl

Correl() renvoie le coefficient de corrélation agrégé pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() permet de déterminer la valeur correspondant au fractile (quantile) inclusif des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() permet de déterminer la valeur correspondant au fractile (quantile) exclusif des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() renvoie le coefficient d'aplatissement des données dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() renvoie la valeur b (segment sur l'axe y) agrégée d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées représentées par des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() renvoie les degrés de liberté agrégés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Cette fonction de script renvoie la statistique F agrégée ($r^2/(1-r^2)$) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() renvoie la valeur m (pente) agrégée d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() renvoie la valeur r^2 agrégée (coefficient de détermination) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() renvoie l'erreur type agrégée de la valeur b d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() renvoie l'erreur type agrégée de la valeur m d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() renvoie l'erreur type agrégée de l'estimation y d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() renvoie la somme de régression agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() renvoie la somme résiduelle agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() renvoie la valeur médiane agrégée des valeurs contenues dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
Median (expression)
```

Skew

Skew() renvoie l'asymétrie de l'expression sur un nombre donné d'enregistrements définis par une clause **group by**.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() renvoie l'écart type des valeurs fournies par l'expression sur un nombre d'enregistrements définis par une clause **group by**.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() renvoie l'erreur type agrégée (stdev/sqrt(n)) pour une série de valeurs que représente l'expression itérée sur un nombre donné d'enregistrements définis par une clause **group by**.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX() renvoie l'erreur type agrégée de la valeur y prévue pour chaque valeur x dans la régression pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

```
STEYX  (y-expression, x-expression)
```

Fonctions d'agrégation statistiques utilisées dans les expressions de graphique

Les fonctions d'agrégation statistiques suivantes peuvent s'utiliser dans les graphiques.

Avg

Avg() renvoie la moyenne agrégée de l'expression ou du champ itéré(e) sur les dimensions du graphique.

```
Avg - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

Correl

Correl() renvoie le coefficient de corrélation agrégé pour deux ensembles de données. La fonction de corrélation mesure la relation établie entre les ensembles de données ; elle est agrégée pour les paires de valeurs (x,y) itérées sur les dimensions du graphique.

```
Correl - fonction de graphique({[SetExpression] [TOTAL [<fld {, fld}>]]}
value1, value2 )
```

Fractile

Fractile() permet de déterminer la valeur correspondant au fractile (quantile) inclusif des données agrégées dans la plage fournie par l'expression itérée sur les dimensions du graphique.

```
Fractile - fonction de graphique({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

FractileExc

FractileExc() permet de déterminer la valeur correspondant au fractile (quantile) exclusif des données agrégées dans la plage fournie par l'expression itérée sur les dimensions du graphique.

```
FractileExc - fonction de graphique({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Kurtosis() permet de déterminer le coefficient d'aplatissement de la plage de données agrégées dans l'expression ou le champ itéré(e) sur les dimensions du graphique.

```
Kurtosis - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

LINEST_B() renvoie la valeur b agrégée (segment sur l'axe y) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_R2 - fonction de graphique({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() renvoie les degrés de liberté agrégés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_DF - fonction de graphique({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() renvoie la statistique F agrégée (r2/(1-r2)) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_F - fonction de graphique({[SetExpression] [TOTAL[<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() renvoie la valeur m agrégée (la pente) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_M - fonction de graphique({[SetExpression] [TOTAL[<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() renvoie la valeur r2 agrégée (le coefficient de détermination) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_R2 - fonction de graphique({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() renvoie l'erreur type agrégée de la valeur b d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_SEB - fonction de graphique({[SetExpression] [TOTAL [<fld{ ,fld}>]]
}y_value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() renvoie l'erreur type agrégée de la valeur m d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_SEM - fonction de graphique([{set_expression}][ distinct ] [total
[<fld {,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() renvoie l'erreur type agrégée de l'estimation y d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_SEY - fonction de graphique({[SetExpression] [TOTAL [<fld{ ,fld}>]]
}y_value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() renvoie la somme de régression agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

```
LINEST_SSREG - fonction de graphique({[SetExpression] [TOTAL [<fld{ ,fld}>]]
}y_value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() renvoie la somme résiduelle agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

LINEST_SSRESID - fonction de graphiqueLINEST_SSRESID() renvoie la somme résiduelle agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par x_value et y_value, itérées sur les dimensions du graphique. LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]]) numérique ArgumentsArgumentDescriptiony_valueExpression ou champ contenant la plage de valeurs y à mesurer.x_valueExpression ou champ contenant la plage de valeurs x à mesurer.y0, x0Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira. SetExpressionPar défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles. DISTINCTSi le terme DISTINCT précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés. TOTALSi le terme TOTAL précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant TOTAL [<fld {.fld}>], où le qualificateur TOTAL est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur TOTAL. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée Aggr en combinaison avec une dimension spécifiée. Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées. An example of how to use linest functionsavg({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]])**

Median

Median() renvoie la valeur médiane de la plage de valeurs agrégées dans l'expression itérée sur les dimensions du graphique.

Median - fonction de graphique({[SetExpression] [**TOTAL** [<fld{, fld}>]]} expr**)**

MutualInfo

MutualInfo calcule les informations mutuelles (MI) entre deux champs ou entre des valeurs agrégées dans **Aggr()**.

```
MutualInfo - fonction de graphique (page 440){[SetExpression] [DISTINCT]
[TOTAL target, driver [, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew() renvoie l'asymétrie agrégée de l'expression ou du champ itéré(e) sur les dimensions du graphique.

```
Skew - fonction de graphique{[SetExpression] [DISTINCT] [TOTAL [<fld{
,fld}>]]} expr)
```

Stdev

Stdev() permet de déterminer l'écart type de la plage de données agrégées dans l'expression ou le champ itéré(e) sur les dimensions du graphique.

```
Stdev - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

Sterr

Sterr() permet de déterminer la valeur de l'erreur type de la moyenne, (stdev/sqrt(n)), pour la série de valeurs agrégées dans l'expression itérée sur les dimensions du graphique.

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

STEYX

STEYX() renvoie l'erreur type agrégée lors de l'estimation des valeurs y pour chaque valeur x dans une régression linéaire définie par une série de coordonnées que représentent des nombres appariés fournis par les expressions **y_value** et **x_value**.

```
STEYX - fonction de graphique{[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value)
```

Avg

Avg() permet de déterminer la valeur moyenne des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Syntaxe :

```
Avg([DISTINCT] expr)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
DISTINCT	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat

`Temp:`
`crosstable (Month, Sales) load * inline [`
`Customer|Jan|Feb|Mar||Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec`
`Astrida|46|60|70|13|78|20|45|65|78|12|78|22`
`Betacab|65|56|22|79|12|56|45|24|32|78|55|15`
`Canutility|77|68|34|91|24|68|57|36|44|90|67|27`
`Divadip|36|44|90|67|27|57|68|47|90|80|94`
`] (delimiter is '|');`
 	
`Avg1:`
`LOAD Customer, Avg(Sales) as MyAverageSalesByCustomer`
`Resident Temp Group By Customer;`
	`Customer`
	`MyAverageSalesByCustomer`
	`Astrida 48.916667`
	`Betacab 44.916667`
	`Canutility 56.916667`
	`Divadip 63.083333`
	Vous pouvez vérifier ce résultat sur la feuille en créant une table comprenant la mesure :
	`Sum(Sales)/12`

Supposons que la table **Temp** est chargée comme dans l'exemple précédent :
`LOAD Customer,Avg(DISTINCT Sales) as MyAvgSalesDistinct`
`Resident Temp Group By Customer;`
	`Customer`
	`MyAverageSalesByCustomer`
	`Astrida 43.1`
	`Betacab 43.909091`
	`Canutility 55.909091`
	`Divadip 61`
	Seules les valeurs distinctes sont comptabilisées. Divisez le total par le nombre de valeurs qui ne sont pas en double.

Avg - fonction de graphique

Avg() renvoie la moyenne agrégée de l'expression ou du champ itéré(e) sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exemples de fonction

Exemple	Résultat
`Avg(Sales)`	Pour une table dotée de la dimension `Customer` et de la mesure `Avg([Sales])`, si les valeurs **Totals** sont affichées, le résultat correspond à 2566.

Exemple	Résultat
Avg([TOTAL (Sales))	53.458333 pour toutes les valeurs de la colonne Customer, puisque le qualificateur TOTAL entraîne la non-prise en compte des dimensions.
Avg (DISTINCT (Sales))	51.862069 pour le total, puisque l'utilisation du qualificateur Distinct permet d'évaluer uniquement les valeurs uniques figurant sous Sales pour chaque Customer.

Données utilisées dans les exemples :

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Aggr - fonction de graphique (page 545)*

Correl

Correl() renvoie le coefficient de corrélation agrégé pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

```
Correl(value1, value2)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value1, value2	Expressions ou champs contenant les deux ensembles d'échantillons pour lesquels le coefficient de corrélation est à mesurer.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat
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;	Dans une table comportant la dimension `Correl_Salary`, le résultat du calcul Correl() dans le script de chargement de données s'affiche de la manière suivante : 0.9270611

Correl - fonction de graphique

Correl() renvoie le coefficient de corrélation agrégé pour deux ensembles de données. La fonction de corrélation mesure la relation établie entre les ensembles de données ; elle est agrégée pour les paires de valeurs (x,y) itérées sur les dimensions du graphique.

Syntaxe :

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value1, value2	Expressions ou champs contenant les deux ensembles d'échantillons pour lesquels le coefficient de corrélation est à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Exemples de fonction

Exemple	Résultat
`Correl (Age, Salary)`	Pour une table dotée de la dimension `Employee name` et de la mesure `Correl(Age, Salary)`, le résultat correspond à 0.9270611. Le résultat s'affiche uniquement pour la cellule des totaux.

Exemple	Résultat
`Correl (TOTAL Age, Salary))`	0.927. Pour une meilleure lisibilité, ce résultat et les résultats suivants sont affichés avec trois décimales. Si vous créez un volet de filtre comportant la dimension Gender et que vous effectuez ensuite des sélections à partir de ce volet, vous obtenez le résultat 0.951 lorsque vous sélectionnez Female et le résultat 0.939 lorsque vous choisissez Male. Ceci s'explique par le fait que la sélection exclut tous les résultats qui ne font pas partie de l'autre valeur de Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Résultat indépendant des sélections effectuées. Ceci s'explique par le fait que l'expression d'ensemble {1} ignore toutes les sélections et toutes les dimensions.
`Correl (TOTAL <Gender> Age, Salary))`	0.927 dans la cellule du total, 0.939 pour toutes les valeurs associées à Male et 0.951 pour toutes les valeurs associées à Female. Ce résultat est obtenu suite aux sélections effectuées dans un volet de filtre basé sur le critère Gender.

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *Aggr - fonction de graphique (page 545)*
p *Avg - fonction de graphique (page 399)*
p *RangeCorrel (page 1338)*

Fractile

Fractile() permet de déterminer la valeur correspondant au fractile (quantile) inclusif des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Vous pouvez utiliser FractileExc (page 410) pour calculer le fractile exclusif.

Syntaxe :

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

Type de données renvoyé : numérique

La fonction renvoie la valeur correspondant au classement tel que défini par `classement = fraction * (N-1) + 1`, où `N` est le nombre de valeurs de `expr`. Si `classement` n'est pas un nombre entier, une interpolation est effectuée entre les deux valeurs les plus proches.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à utiliser lors du calcul du fractile.
fraction	Nombre compris entre 0 et 1 correspondant au fractile (quantile exprimé sous forme de fraction) à calculer.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat
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;	Dans une table comportant les dimensions `Type` et `MyFractile`, les résultats des calculs Fractile() dans le script de chargement de données sont les suivants : Type MyFractile Comparison 27.5 Observation 36

Fractile - fonction de graphique

Fractile() permet de déterminer la valeur correspondant au fractile (quantile) inclusif des données agrégées dans la plage fournie par l'expression itérée sur les dimensions du graphique.

Vous pouvez utiliser FractileExc - fonction de graphique (page 411) pour calculer le fractile exclusif.

Syntaxe :

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

Type de données renvoyé : numérique

La fonction renvoie la valeur correspondant au classement tel que défini par `classement = fraction * (N-1) + 1`, où `N` est le nombre de valeurs de `expr`. Si `classement` n'est pas un nombre entier, une interpolation est effectuée entre les deux valeurs les plus proches.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à utiliser lors du calcul du fractile.
fraction	Nombre compris entre 0 et 1 correspondant au fractile (quantile exprimé sous forme de fraction) à calculer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exemples de fonction

Exemple	Résultat
`Fractile (Sales, 0.75)`	Pour une table incluant la dimension `Customer` et la mesure `Fractile([Sales])`, si les valeurs **Totaux** sont affichées, le résultat correspond à 71,75. Il s'agit du point dans la distribution des valeurs de `Sales` en dessous duquel tombent 75 % des valeurs.
`Fractile (TOTAL Sales, 0.75))`	71,75 pour toutes les valeurs de la colonne `Customer`, puisque le qualificateur TOTAL entraîne la non-prise en compte des dimensions.
`Fractile (DISTINCT Sales, 0.75)`	70 pour le total, puisque l'utilisation du qualificateur DISTINCT permet d'évaluer uniquement les valeurs uniques figurant sous `Sales` pour chaque `Customer`.

Données utilisées dans les exemples :

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Aggr - fonction de graphique (page 545)*

FractileExc

FractileExc() permet de déterminer la valeur correspondant au fractile (quantile) exclusif des données agrégées dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Vous pouvez utiliser Fractile (page 405) pour calculer le fractile inclusif.

Syntaxe :

```
FractileExc(expr, fraction)
```

Type de données renvoyé : numérique

La fonction renvoie la valeur correspondant au classement tel que défini par `classement = fraction * (N+1)`, où `N` est le nombre de valeurs de `expr`. Si `classement` n'est pas un nombre entier, une interpolation est effectuée entre les deux valeurs les plus proches.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à utiliser lors du calcul du fractile.
fraction	Nombre compris entre 0 et 1 correspondant au fractile (quantile exprimé sous forme de fraction) à calculer.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, FractileExc(Value,0.75) as MyFractile Resident Table1 Group By Type;	Dans une table comportant les dimensions `Type` et `MyFractile`, les résultats des calculs FractileExc() dans le script de chargement de données sont les suivants : Type MyFractile Comparison 28.5 Observation 38

FractileExc - fonction de graphique

FractileExc() permet de déterminer la valeur correspondant au fractile (quantile) exclusif des données agrégées dans la plage fournie par l'expression itérée sur les dimensions du graphique.

Vous pouvez utiliser Fractile - fonction de graphique (page 407) pour calculer le fractile inclusif.

Syntaxe :

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

Type de données renvoyé : numérique

La fonction renvoie la valeur correspondant au classement tel que défini par `classement = fraction * (N+1)`, où `N` est le nombre de valeurs de `expr`. Si `classement` n'est pas un nombre entier, une interpolation est effectuée entre les deux valeurs les plus proches.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à utiliser lors du calcul du fractile.
fraction	Nombre compris entre 0 et 1 correspondant au fractile (quantile exprimé sous forme de fraction) à calculer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exemples de fonction

Exemple	Résultat
`FractileExc (Sales, 0.75)`	Pour une table incluant la dimension `Customer` et la mesure `FractileExc([Sales])`, si les valeurs **Totaux** sont affichées, le résultat correspond à 75,25. Il s'agit du point dans la distribution des valeurs de `Sales` en dessous duquel tombent 75 % des valeurs.
`FractileExc (TOTAL Sales, 0.75))`	75,25 pour toutes les valeurs de la colonne `Customer`, puisque le qualificateur TOTAL entraîne la non-prise en compte des dimensions.
`FractileExc (DISTINCT Sales, 0.75)`	73,50 pour le total, puisque l'utilisation du qualificateur DISTINCT permet d'évaluer uniquement les valeurs uniques figurant sous `Sales` pour chaque `Customer`.

Données utilisées dans les exemples :

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Aggr - fonction de graphique (page 545)*

Kurtosis

Kurtosis() renvoie le coefficient d'aplatissement des données dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat
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;	Dans une table comportant les dimensions `Type`, `MyKurtosis1` et `MyKurtosis2`, les résultats des calculs Kurtosis() dans le script de chargement de données sont les suivants : Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - fonction de graphique

Kurtosis() permet de déterminer le coefficient d'aplatissement de la plage de données agrégées dans l'expression ou le champ itéré(e) sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Example table

Type	Value																			
Comparison	2	27	38	31	1	19	1	34	3	1	2	3	2	1	2	1	3	29	37	2
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Exemples de fonction

Exemple	Résultat
`Kurtosis (Value)`	Pour une table dotée de la dimension `Type` et de la mesure `Kurtosis(Value)`, si des **Totaux** ont affichés pour la table et que le formatage des nombres est défini sur 3 chiffres significatifs, le résultat correspond à 1.252. Pour le type `Comparison`, le résultat correspond à 1.161 et pour le type `Observation`, à 1.115.
`Kurtosis (TOTAL Value))`	1.252 pour toutes les valeurs de la colonne `Type`, puisque le qualificateur TOTAL entraîne la non-prise en compte des dimensions.

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *Avg - fonction de graphique (page 399)*

LINEST_B

LINEST_B() renvoie la valeur b (segment sur l'axe y) agrégée d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées représentées par des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_B - fonction de graphique

LINEST_B() renvoie la valeur b agrégée (segment sur l'axe y) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.

Argument	Description
y0_const, x0_const	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {,fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*
p *Avg - fonction de graphique (page 399)*

LINEST_DF

LINEST_DF() renvoie les degrés de liberté agrégés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_DF - fonction de graphique

LINEST_DF() renvoie les degrés de liberté agrégés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_F

Cette fonction de script renvoie la statistique F agrégée ($r^2/(1-r^2)$) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_F - fonction de graphique

LINEST_F() renvoie la statistique F agrégée (r2/(1-r2)) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_M

LINEST_M() renvoie la valeur m (pente) agrégée d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_M - fonction de graphique

LINEST_M() renvoie la valeur m agrégée (la pente) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_R2

LINEST_R2() renvoie la valeur r^2 agrégée (coefficient de détermination) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_R2 - fonction de graphique

LINEST_R2() renvoie la valeur r2 agrégée (le coefficient de détermination) d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_SEB

LINEST_SEB() renvoie l'erreur type agrégée de la valeur b d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_SEB - fonction de graphique

LINEST_SEB() renvoie l'erreur type agrégée de la valeur b d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_SEM

LINEST_SEM() renvoie l'erreur type agrégée de la valeur m d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_SEM - fonction de graphique

LINEST_SEM() renvoie l'erreur type agrégée de la valeur m d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_SEY

LINEST_SEY() renvoie l'erreur type agrégée de l'estimation y d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_SEY - fonction de graphique

LINEST_SEY() renvoie l'erreur type agrégée de l'estimation y d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_SSREG

LINEST_SSREG() renvoie la somme de régression agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_SSREG - fonction de graphique

LINEST_SSREG() renvoie la somme de régression agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés fournis par les expressions **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

p *Avg - fonction de graphique (page 399)*

LINEST_SSRESID

LINEST_SSRESID() renvoie la somme résiduelle agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y(0), x(0)	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*

LINEST_SSRESID - fonction de graphique

LINEST_SSRESID() renvoie la somme résiduelle agrégée des carrés d'une régression linéaire définie par l'équation y=mx+b pour une série de coordonnées que représentent des nombres appariés dans les expressions fournies par **x_value** et **y_value**, itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.
y0, x0	Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique. *À moins que les coordonnées y0 et x0 soient toutes deux définies, la fonction exige au moins deux paires de données valides pour effectuer le calcul. Si y0 et x0 sont définies, une seule paire de données suffira.*
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Il est possible de définir une valeur y0 facultative pour forcer la ligne de régression à passer par l'axe des ordonnées en un point donné. En définissant à la fois y0 et x0, il est possible de forcer la ligne de régression à passer par une coordonnée fixe unique.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Voir aussi :

p *Exemples d'utilisation des fonctions linest (page 458)*
p *Avg - fonction de graphique (page 399)*

Median

Median() renvoie la valeur médiane agrégée des valeurs contenues dans l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Syntaxe :

```
Median (expr)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Exemple : Expression de script via une médiane

Exemple - expression de script

Script de chargement

Chargez l'expression de script et les données inline suivantes dans l'éditeur de chargement de données pour cet exemple.

```
Table 1: Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8
B,9];                                    Median: LOAD Letter, Median(Number) as MyMedian Resident Table1 Grou
```

Création d'une visualisation

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Letter** et **MyMedian**.

Résultat

Letter	MyMedian
A	3.5
B	8

Explication

La médiane est considérée comme le nombre du "milieu" lorsque les nombres ont été triés dans l'ordre croissant. Si l'ensemble de données comporte un nombre pair de valeurs, la fonction renvoie la moyenne des deux valeurs du milieu. Dans cet exemple, la médiane est calculée pour chaque ensemble de valeurs de **A** et de **B**, qui sont 3.5 et 8, respectivement.

Median - fonction de graphique

Median() renvoie la valeur médiane de la plage de valeurs agrégées dans l'expression itérée sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemple : Expression de graphique via une médiane

Exemple - expression de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple d'expression de graphique ci-dessous.

```
Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8 B,9];
```

Création d'une visualisation

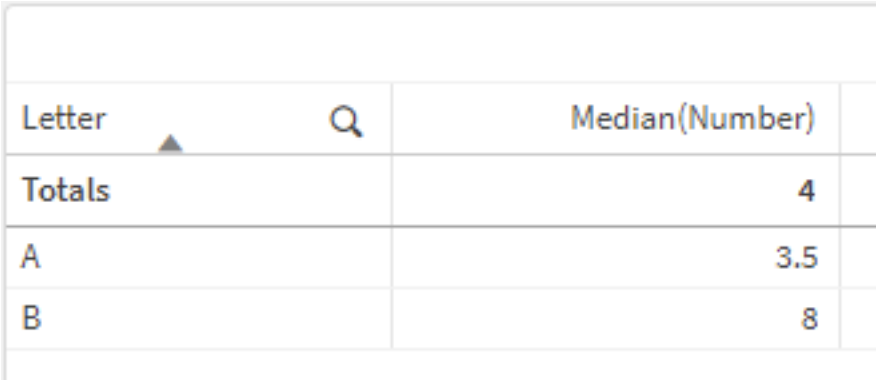
Créez une visualisation de table dans une feuille Qlik Sense avec **Letter** comme dimension.

Expression de graphique

Ajoutez à la table l'expression suivante, sous forme de mesure :

```
Median(Number)
```

Résultat



Letter	Median(Number)
Totals	4
A	3.5
B	8

Explication

La médiane est considérée comme le nombre du "milieu" lorsque les nombres ont été triés dans l'ordre croissant. Si l'ensemble de données comporte un nombre pair de valeurs, la fonction renvoie la moyenne des deux valeurs du milieu. Dans cet exemple, la médiane est calculée pour chaque ensemble de valeurs de **A** et de **B**, qui sont 3.5 et 8, respectivement.

La médiane de **Totals** est calculée à partir de l'ensemble des valeurs, à savoir, 4.

Voir aussi :

p *Avg - fonction de graphique (page 399)*

MutualInfo - fonction de graphique

MutualInfo calcule les informations mutuelles (MI) entre deux champs ou entre des valeurs agrégées dans **Aggr()**.

MutualInfo renvoie les informations mutuelles agrégées pour deux ensembles de données. Cela permet l'analyse de pilote clé entre un champ et un pilote potentiel. La fonction Informations mutuelles mesure la relation établie entre les ensembles de données ; elle est agrégée pour les paires de valeurs (x,y) itérées sur les dimensions du graphique. La fonction Informations mutuelles est mesurée entre 0 et 1 et peut être formatée comme une valeur de centile. **MutualInfo** est défini par des sélections ou une expression d'ensemble.

MutualInfo permet différents types d'analyse MI :

- Pair-wise MI : Calcule la valeur MI entre un champ pilote et un champ cible.
- Driver breakdown by value : La valeur MI est calculée entre des valeurs de champ individuelles des champs pilote et cible.
- Feature selection : Utilisez **MutualInfo** dans des bulles pour générer une matrice dans laquelle tous les champs sont comparés les uns aux autres en fonction de la valeur MI.

MutualInfo n'indique pas forcément la causalité entre les champs partageant des informations mutuelles. Deux champs peuvent partager des informations mutuelles sans être des pilotes égaux l'un pour l'autre. Par exemple, lors de la comparaison entre les ventes de crème glacée et la température extérieure, **MutualInfo** affichera les informations mutuelles entre les deux. Cela n'indiquera pas si c'est la température extérieure qui dirige les ventes de crème glacée, ce qui est probable, ou si ce sont les ventes de crème glacée qui dirigent la température extérieure, ce qui est peu probable.

Lors du calcul des informations mutuelles, les associations affectent la correspondance entre les valeurs des champs provenant de différentes tables et leur fréquence.

Les valeurs renvoyées pour les mêmes champs ou sélections peuvent légèrement varier. Cela est dû au fait que chaque appel **MutualInfo** agit sur un échantillon sélectionné de manière aléatoire et à la nature aléatoire inhérente de l'algorithme **MutualInfo**.

MutualInfo peut être appliqué à la fonction **Aggr()**.

Syntaxe :

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
field1, field2	Expressions ou champs contenant les deux ensembles d'échantillons pour lesquels les informations mutuelles doivent être mesurées.
datatype	Types de données contenus dans la cible et le pilote, 1 ou 'dd' pour discret:discret 2 ou 'cc' pour continu:continu 3 ou 'cd' pour continu:discret 4 ou 'dc' pour discret:continu Les types de données ne sont pas sensibles à la casse.
breakdownbyvalue	Valeur statique correspondant à une valeur du pilote. Si cette valeur est fournie, le calcul calcule la contribution MI pour cette valeur. Vous pouvez utiliser **ValueList()** ou **ValueLoop()**. Si **Null()** est ajouté, le calcul calcule la valeur MI globale pour toutes les valeurs du pilote. Pour la répartition par valeur, le pilote doit contenir des données discrètes.

Argument	Description
samplesize	Nombre de valeurs à échantillonner de la cible et du pilote. L'échantillonnage est aléatoire. **MutualInfo** nécessite une taille d'échantillon minimale de 80. Par défaut, les échantillons exclusivement **MutualInfo** contenant jusqu'à 10 000 paires de données tels que **MutualInfo** peuvent consommer beaucoup de ressources. Vous pouvez spécifier des nombres de paires de données supérieurs dans la taille d'échantillon. En cas d'expiration de **MutualInfo**, réduisez la taille de l'échantillon.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Exemples de fonction

Exemple	Résultat
`mutualinfo (Age, Salary, 1)`	Pour une table dotée de la dimension `Employee name` et de la mesure `mutualinfo(Age, Salary, 1)`, le résultat correspond à 0.99820986. Le résultat s'affiche uniquement pour la cellule des totaux.
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	Si vous créez un volet de filtre comportant la dimension Gender et que vous effectuez ensuite des sélections à partir de ce volet, vous obtenez le résultat 0.99805677 lorsque vous sélectionnez Female et le résultat 0.99847373 lorsque vous sélectionnez Male. Ceci s'explique par le fait que la sélection exclut tous les résultats qui ne font pas partie de l'autre valeur de Gender.

Exemple	Résultat
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. La sélection de n'importe quelle valeur de Gender remplace cela par 0.
`mutualinfo ({1} TOTAL Age, Salary, 1, null())`	0.99820986. Cela est indépendant des sélections effectuées. L'expression d'ensemble {1} ignore toutes les sélections et toutes les dimensions.

Données utilisées dans les exemples :

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary

Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000

Celine Gagnon|48|Female|38000

Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000
```

```
David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000

Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000

Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000

John Doe|36|Male|59000

John Lemon|43|Male|21000

Karen Helmkey|54|Female|25000

Karl Berger|38|Male|68000

Karl Straubaum|30|Male|40000

Kaya Alpan|32|Female|60000

Kenneth Finley|21|Male|25000
```

```
Leif Shine|63|Male|70000

Lennart Skoglund|63|Male|24000

Leona Korhonen|46|Female|50000

Lina André|50|Female|65000

Louis Presley|29|Male|36000

Luke Langston|50|Male|63000

Marcus Salvatori|31|Male|46000

Marie Simon|57|Female|23000

Mario Rossi|39|Male|62000

Markus Danzig|26|Male|48000

Michael Carlen|21|Male|45000

Michelle Tyson|44|Female|69000

Mike Ashkenaz|45|Male|68000

Miro Ito|40|Male|39000

Nina Mihn|62|Female|57000

Olivia Nguyen|35|Female|51000

Olivier Simenon|44|Male|31000

Östen Ärlig|68|Male|57000

Pamala Garcia|69|Female|29000

Paolo Romano|34|Male|45000

Pat Taylor|67|Female|69000

Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000
```

```
Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000

Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000

Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

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

Skew

Skew() renvoie l'asymétrie de l'expression sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
DISTINCT	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `Type` et *MySkew* comme dimensions.

Données résultantes

Exemple	Résultat

`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;`
	Les résultats du calcul Skew() sont les suivants :
	• `Type` correspond à `MySkew`
	• `Comparison` correspond à 0.86414768
	• `Observation` correspond à 0.32625351

Skew - fonction de graphique

Skew() renvoie l'asymétrie agrégée de l'expression ou du champ itéré(e) sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `Type` comme dimension et `Skew(Value)` comme mesure.

Assurez-vous que `Totals` est activé dans les propriétés de la table.

Exemple	Résultat
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 '|');	Les résultats du calcul Skew(Value) sont les suivants : • Totalcorrespond à 0.23522195 • Comparison correspond à 0.86414768 • Observation correspond à 0.32625351

Voir aussi :

p *Avg - fonction de graphique (page 399)*

Stdev

Stdev() renvoie l'écart type des valeurs fournies par l'expression sur un nombre d'enregistrements définis par une clause **group by**.

Syntaxe :

```
Stdev([distinct] expr)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `Type` et `MyStdev` comme dimensions.

Données résultantes

Exemple	Résultat

`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;`
	Les résultats du calcul Stdev() sont les suivants :
	• `Type` correspond à `MyStdev`
	• `Comparison` correspond à 14.61245
	• `Observation` correspond à 12.507997

Stdev - fonction de graphique

Stdev() permet de déterminer l'écart type de la plage de données agrégées dans l'expression ou le champ itéré(e) sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `Type` comme dimension et `Stdev(Value)` comme mesure.

Assurez-vous que `Totals` est activé dans les propriétés de la table.

Exemple	Résultat

`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`
`'|');`
	Les résultats du calcul Stdev(Value) sont les suivants :
	• `Total`correspond à 15.47529
	• `Comparison` correspond à 14.61245
	• `Observation` correspond à 12.507997

Voir aussi :

p *Avg - fonction de graphique (page 399)*
p *STEYX - fonction de graphique (page 456)*

Sterr

Sterr() renvoie l'erreur type agrégée (stdev/sqrt(n)) pour une série de valeurs que représente l'expression itérée sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat

`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;`
	Dans une table comportant les dimensions `Type` et `MySterr`, les résultats du calcul Sterr() dans le script de chargement de données sont les suivants :
	
	`Type MySterr`
	`Comparison 3.2674431`
	`Observation 2.7968733`

Sterr - fonction de graphique

Sterr() permet de déterminer la valeur de l'erreur type de la moyenne, (stdev/sqrt(n)), pour la série de valeurs agrégées dans l'expression itérée sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `Type` comme dimension et `Sterr(Value)` comme mesure.

Assurez-vous que `Totals` est activé dans les propriétés de la table.

Exemple	Résultat
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 '|');	Les résultats du calcul Sterr(Value) sont les suivants : • Totalcorrespond à 2.4468583 • Comparison correspond à 3.2674431 • Observation correspond à 2.7968733

Voir aussi :

p *Avg - fonction de graphique (page 399)*
p *STEYX - fonction de graphique (page 456)*

STEYX

STEYX() renvoie l'erreur type agrégée de la valeur y prévue pour chaque valeur x dans la régression pour une série de coordonnées que représentent des nombres appariés dans les expressions x-expression et y-expression itérées sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

```
STEYX (y_value, x_value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x à mesurer.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat

`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;`
	Dans une table comportant la dimension `MySTEYX`, le résultat du calcul STEYX() dans le script de chargement de données est égal à 2.0714764.

STEYX - fonction de graphique

STEYX() renvoie l'erreur type agrégée lors de l'estimation des valeurs y pour chaque valeur x dans une régression linéaire définie par une série de coordonnées que représentent des nombres appariés fournis par les expressions **y_value** et **x_value**.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
y_value	Expression ou champ contenant la plage de valeurs y connues à mesurer.
x_value	Expression ou champ contenant la plage de valeurs x connues à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {,fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Le paramètre de la fonction d'agrégation ne doit pas comprendre d'autres fonctions d'agrégation, à moins que ces agrégations internes ne contiennent le qualificateur **TOTAL**. Pour des agrégations imbriquées plus complexes, utilisez la fonction avancée **Aggr** en combinaison avec une dimension spécifiée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes dans une ou les deux paires de données sont ignorées.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Créez ensuite un tableau simple en utilisant `KnownY` et `KnownX` comme dimensions et `Steyx(KnownY,KnownX)` comme mesure.

Assurez-vous que `Totals` est activé dans les propriétés de la table.

Exemple	Résultat
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 '|');	Le résultat du calcul STEYX(KnownY,KnownX) correspond à 2.071 (si le formatage des nombres est défini sur 3 décimales.)

Voir aussi :

p *Avg - fonction de graphique (page 399)*
p *Sterr - fonction de graphique (page 453)*

Exemples d'utilisation des fonctions linest

Les fonctions linest permettent de déterminer les valeurs associées à une analyse de type régression linéaire. Cette section décrit la procédure de création de visualisations à l'aide d'échantillons de données dans le but d'identifier les valeurs des fonctions linest disponibles dans Qlik Sense. Les fonctions linest s'utilisent aussi bien dans le script de chargement de données que dans les expressions de graphique.

Pour une description de la syntaxe et des arguments, voir les rubriques des différentes fonctions de graphique et de script linest.

Données et expressions de script utilisées dans les exemples

Chargez les expressions de script et les données inline suivantes dans l'éditeur de chargement de données pour créer les exemples d'expression linest() ci-dessous.

```
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;
```

Exemple 1 : Expressions de script utilisant linest

Exemple : Expressions de script

Création d'une visualisation à partir de calculs de script de chargement de données

Créez la visualisation d'une table dans une feuille Qlik Sense avec les champs suivants sous forme de colonnes :

- Linest_B
- Linest_DF
- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Résultat

La table contenant les résultats des calculs linest réalisés dans le script de chargement de données devrait avoir l'aspect suivant :

Table des résultats

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

Table des résultats

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Exemple 2 : Expressions de graphique utilisant linest

Exemple : Expressions de graphique

Créez la visualisation d'une table dans une feuille Qlik Sense avec les champs suivants sous forme de dimensions :

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Cette expression utilise la fonction synthetic dimensions pour créer les étiquettes des dimensions en utilisant les noms des fonctions linest. Pour gagner de l'espace, vous pouvez renommer l'étiquette en **Linest functions**.

Ajoutez à la table l'expression suivante en tant que mesure :

```
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) )
```

Cette expression affiche la valeur du résultat de chaque fonction linest par rapport au nom correspondant dans la dimension synthétique. Le résultat de `Linest_b(Y,X)` s'affiche en regard de **linest_b** et ainsi de suite.

Résultat

Table des résultats

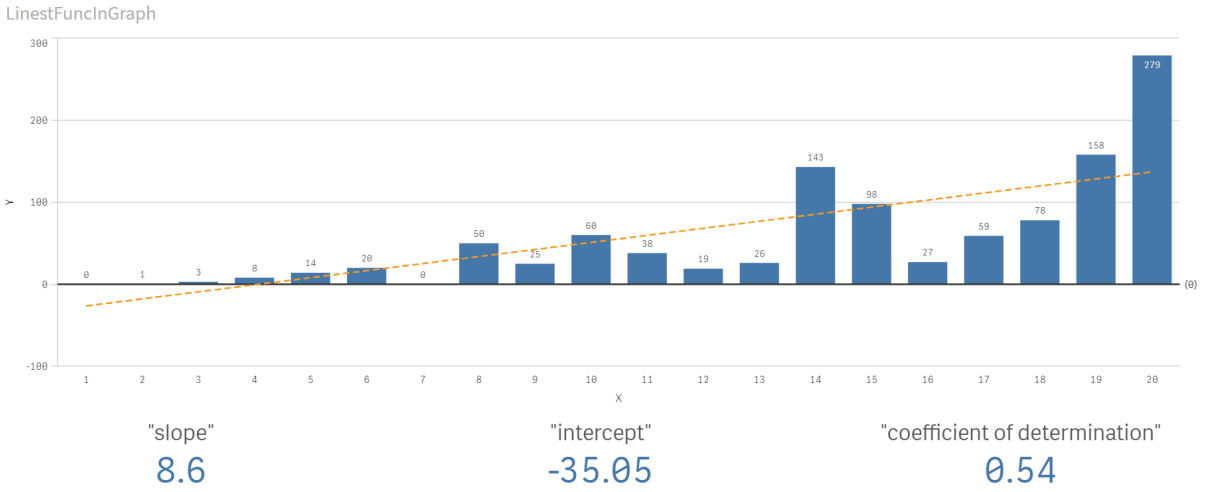
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Exemple 3 : Expressions de graphique utilisant linest

Exemple : Expressions de graphique

1. Créez une visualisation de graphique en barres dans une feuille Qlik Sense avec **X** comme dimension et **Y** comme mesure.
2. Ajoutez une courbe de tendance linéaire à la mesure Y.
3. Ajoutez une visualisation d'indicateur KPI à la feuille.
    1. Ajoutez une *pente* comme étiquette pour l'indicateur KPI.
    2. Ajoutez `sum(Linest_M)` comme expression pour l'indicateur KPI.
4. Ajoutez une deuxième visualisation d'indicateur KPI à la feuille.
    1. Ajoutez *intercept* comme étiquette pour l'indicateur KPI.
    2. Ajoutez `Sum(Linest_B)` comme expression pour l'indicateur KPI.
5. Ajoutez une troisième visualisation d'indicateur KPI à la feuille.
    1. Ajoutez *coefficient of determination* comme étiquette pour l'indicateur KPI.
    2. Ajoutez `Sum(Linest_R2)` comme expression pour l'indicateur KPI.

Résultat



Explication

Le graphique en barres affiche le tracé des données X et Y. Les fonctions linest() pertinentes fournissent des valeurs pour l'équation de régression linéaire sur laquelle est basée la courbe de tendance, à savoir y = m * x + b. L'équation utilise la méthode des "moindres carrés" pour calculer une ligne droite (courbe de tendance) en renvoyant un tableau qui décrit une ligne la mieux adaptée aux données.

Les indicateurs KPI affichent les résultats des fonctions linest() **sum(Linest_M)** pour la pente et **sum (Linest_B)** pour l'interception Y, qui sont des variables de l'équation de régression linéaire, et la valeur R2 agrégée correspondante pour le coefficient de détermination.

Fonctions de test statistique

Les fonctions de test statistiques s'utilisent à la fois dans le script de chargement de données et dans les expressions de graphique, même si leur syntaxe diffère.

Fonctions chi2-test

S'utilise généralement dans l'étude de variables qualitatives. Il est possible de comparer les fréquences observées dans une table de fréquences unidirectionnelle aux fréquences attendues ou d'étudier la connexion entre deux variables d'un tableau de contingence.

Fonctions t-test

Les fonctions de test t s'utilisent dans l'examen statistique de deux populations moyennes. Un test t portant sur deux échantillons examine si deux échantillons sont différents. Ce test s'emploie fréquemment lorsque deux distributions normales présentent des variances inconnues et lorsqu'une expérience utilise une petite taille d'échantillon.

Fonctions z-test

Examen statistique de deux populations moyennes. Un test z portant sur deux échantillons examine si deux échantillons sont différents. Ce test s'emploie fréquemment lorsque deux distributions normales présentent des variances connues et lorsqu'une expérience utilise une grande taille d'échantillon.

Fonctions chi2-test

S'utilise généralement dans l'étude de variables qualitatives. Il est possible de comparer les fréquences observées dans une table de fréquences unidirectionnelle aux fréquences attendues ou d'étudier la connexion entre deux variables d'un tableau de contingence.Chi-squared test functions are used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more groups. Often a histogram is used, and the different bins are compared to an expected distribution.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Chi2Test_chi2
Chi2Test_chi2() renvoie la valeur agrégée de test chi^2 pour une ou deux séries de valeurs.

```
Chi2Test_chi2() renvoie la valeur agrégée de test chi2 pour une ou deux séries de valeurs.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df() renvoie la valeur df agrégée (degrés de liberté) de test chi^2 pour une ou deux séries de valeurs.

```
Chi2Test_df() renvoie la valeur df agrégée (degrés de liberté) de test chi2 pour une ou deux séries de valeurs.(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p() renvoie la valeur p agrégée (précision) de test chi^2 pour une ou deux séries de valeurs.

```
Chi2Test_p - fonction de graphique(col, row, actual_value[, expected_value])
```

Voir aussi :

p *Fonctions t-test (page 465)*
p *Fonctions z-test (page 502)*

Chi2Test_chi2

Chi2Test_chi2() renvoie la valeur agrégée de test chi^2 pour une ou deux séries de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Toutes les fonctions Qlik Sense chi^2 -test comportent les mêmes arguments.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
col, row	Colonne et ligne spécifiées dans la matrice de valeurs en cours de test.
actual_value	Valeur observée pour les données définies à la colonne et à la ligne spécifiées par les arguments **col** et **row**.
expected_ value	Valeur attendue pour la distribution à la colonne et à la ligne spécifiées par les arguments **col** et **row**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Voir aussi :

p *Exemples d'utilisation des fonctions chi2-test dans les graphiques (page 518)*
p *Exemples d'utilisation des fonctions chi2-test dans le script de chargement de données (page 522)*

Chi2Test_df

Chi2Test_df() renvoie la valeur df agrégée (degrés de liberté) de test chi^2 pour une ou deux séries de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Toutes les fonctions Qlik Sense chi^2 -test comportent les mêmes arguments.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
col, row	Colonne et ligne spécifiées dans la matrice de valeurs en cours de test.
actual_value	Valeur observée pour les données définies à la colonne et à la ligne spécifiées par les arguments **col** et **row**.
expected_value	Valeur attendue pour la distribution à la colonne et à la ligne spécifiées par les arguments **col** et **row**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Voir aussi :

p *Exemples d'utilisation des fonctions chi2-test dans les graphiques (page 518)*
p *Exemples d'utilisation des fonctions chi2-test dans le script de chargement de données (page 522)*

Chi2Test_p - fonction de graphique

Chi2Test_p() renvoie la valeur p agrégée (précision) de test chi^2 pour une ou deux séries de valeurs. Il est possible d'exécuter le test sur les valeurs figurant dans **actual_value** afin de rechercher les variations dans la matrice spécifiée par **col** et **row** ou de comparer les valeurs figurant dans **actual_value** aux valeurs correspondantes dans **expected_value** (si spécifiées).

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Toutes les fonctions Qlik Sense chi^2 -test comportent les mêmes arguments.

Syntaxe :

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
col, row	Colonne et ligne spécifiées dans la matrice de valeurs en cours de test.
actual_value	Valeur observée pour les données définies à la colonne et à la ligne spécifiées par les arguments **col** et **row**.
expected_ value	Valeur attendue pour la distribution à la colonne et à la ligne spécifiées par les arguments **col** et **row**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Voir aussi :

p *Exemples d'utilisation des fonctions chi2-test dans les graphiques (page 518)*
p *Exemples d'utilisation des fonctions chi2-test dans le script de chargement de données (page 522)*

Fonctions t-test

Les fonctions de test t s'utilisent dans l'examen statistique de deux populations moyennes. Un test t portant sur deux échantillons examine si deux échantillons sont différents. Ce test s'emploie fréquemment lorsque deux distributions normales présentent des variances inconnues et lorsqu'une expérience utilise une petite taille d'échantillon.

Dans les sections qui suivent, les fonctions de test statistique t-test sont groupées d'après le test d'échantillons d'étudiants applicable à chaque type de fonction.

Création d'un rapport t-test type (page 523)

Application de t-tests à deux échantillons indépendants

Les fonctions suivantes s'appliquent à des t-tests de deux échantillons d'étudiants indépendants.

ttest_conf

TTest_conf renvoie la valeur agrégée de l'intervalle de confiance du test t pour deux échantillons indépendants.

```
TTest_conf renvoie la valeur agrégée de l'intervalle de confiance du test t
pour deux échantillons indépendants. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() renvoie la valeur agrégée (degrés de liberté) du test t d'étudiants pour deux séries de valeurs indépendantes.

```
TTest_df() renvoie la valeur agrégée (degrés de liberté) du test t
d'étudiants pour deux séries de valeurs indépendantes. (grp, value [, eq_var)
```

ttest_dif

TTest_dif() est une fonction numérique qui renvoie la différence moyenne agrégée de test t d'étudiants pour deux séries de valeurs indépendantes.

```
TTest_dif() est une fonction numérique qui renvoie la différence moyenne
agrégée de test t d'étudiants pour deux séries de valeurs indépendantes.
(grp, value)
```

ttest_lower

TTest_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
TTest_lower() renvoie la valeur agrégée de la borne inférieure de
l'intervalle de confiance pour deux séries de valeurs indépendantes. (grp,
value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour deux séries de valeurs indépendantes.

```
TTest_sig() renvoie le niveau de précision bilatéral de test t agrégé
d'étudiants pour deux séries de valeurs indépendantes. (grp, value [, eq_
var])
```

ttest_sterr

TTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour deux séries de valeurs indépendantes.

```
TTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de
test t d'étudiants pour deux séries de valeurs indépendantes. (grp, value [,
eq_var])
```

ttest_t

TTest_t() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

```
TTest_t() renvoie la valeur t agrégée pour deux séries indépendantes de
valeurs. (grp, value [, eq_var])
```

ttest_upper

TTest_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
TTest_upper() renvoie la valeur agrégée de la borne supérieure de
l'intervalle de confiance pour deux séries de valeurs indépendantes. (grp,
value [, sig [, eq_var]])
```

Application de t-tests à deux échantillons pondérés indépendants

Les fonctions suivantes s'appliquent à des t-tests de deux échantillons d'étudiants indépendants où la série de données d'entrée est fournie dans un format bicolonne pondéré.

ttestw_conf

TTestw_conf() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

```
TTestw_conf() renvoie la valeur t agrégée pour deux séries indépendantes de
valeurs. (weight, grp, value [, sig[, eq_var]])
```

ttestw_df

TTestw_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour deux séries de valeurs indépendantes.

```
TTestw_df() renvoie la valeur df (degrés de liberté) agrégée du test t
d'étudiants pour deux séries de valeurs indépendantes. (weight, grp, value [,
eq_var])
```

ttestw_dif

TTestw_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour deux séries de valeurs indépendantes.

```
TTestw_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour
deux séries de valeurs indépendantes. ( weight, grp, value)
```

ttestw_lower

TTestw_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
TTestw_lower() renvoie la valeur agrégée de la borne inférieure de
l'intervalle de confiance pour deux séries de valeurs indépendantes. (weight,
grp, value [, sig[, eq_var]])
```

ttestw_sig

TTestw_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour deux séries de valeurs indépendantes.

TTestw_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour deux séries de valeurs indépendantes. (weight, grp, value [, eq_var]**)**

ttestw_sterr

TTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour deux séries de valeurs indépendantes.

TTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour deux séries de valeurs indépendantes. (weight, grp, value [, eq_var]**)**

ttestw_t

TTestw_t() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

TTestw_t() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs. (weight, grp, value [, eq_var]**)**

ttestw_upper

TTestw_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

TTestw_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes. (weight, grp, value [, sig [, eq_var]]**)**

Application de t-tests à un échantillon

Les fonctions suivantes s'appliquent à des t-tests d'un seul échantillon d'étudiants.

ttest1_conf

TTest1_conf() renvoie la valeur agrégée de l'intervalle de confiance pour une série de valeurs.

TTest1_conf() renvoie la valeur agrégée de l'intervalle de confiance pour une série de valeurs. (value [, sig]**)**

ttest1_df

TTest1_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour une série de valeurs.

TTest1_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour une série de valeurs. (value**)**

ttest1_dif

TTest1_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour une série de valeurs.

TTest1_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour une série de valeurs. (value**)**

ttest1_lower

TTest1_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour une série de valeurs.

```
TTest1_lower() renvoie la valeur agrégée de la borne inférieure de
l'intervalle de confiance pour une série de valeurs. (value [, sig])
```

ttest1_sig

TTest1_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour une série de valeurs.

```
TTest1_sig() renvoie le niveau de précision bilatéral de test t agrégé
d'étudiants pour une série de valeurs. (value)
```

ttest1_sterr

TTest1_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour une série de valeurs.

```
TTest1_sterr() renvoie l'erreur type agrégée de la différence moyenne de
test t d'étudiants pour une série de valeurs. (value)
```

ttest1_t

TTest1_t() renvoie la valeur t agrégée pour une série de valeurs.

```
TTest1_t() renvoie la valeur t agrégée pour une série de valeurs. (value)
```

ttest1_upper

TTest1_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour une série de valeurs.

```
TTest1_upper() renvoie la valeur agrégée de la borne supérieure de
l'intervalle de confiance pour une série de valeurs. (value [, sig])
```

Application de t-tests à un échantillon pondéré

Les fonctions suivantes s'appliquent à des t-tests d'un seul échantillon d'étudiants où la série de données d'entrée est fournie au format bicolonne pondéré.

ttest1w_conf

TTest1w_conf() est une fonction **numérique** qui renvoie la valeur agrégée de l'intervalle de confiance pour une série de valeurs.

```
TTest1w_conf() est une fonction numérique qui renvoie la valeur agrégée de
l'intervalle de confiance pour une série de valeurs. (weight, value [, sig])
```

ttest1w_df

TTest1w_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour une série de valeurs.

```
TTest1w_df() renvoie la valeur df (degrés de liberté) agrégée du test t
d'étudiants pour une série de valeurs. (weight, value)
```

ttest1w_dif

TTest1w_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour une série de valeurs.

```
TTest1w_dif() renvoie la différence moyenne agrégée de test t d'étudiants
pour une série de valeurs. (weight, value)
```

ttest1w_lower

TTest1w_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour une série de valeurs.

```
TTest1w_lower() renvoie la valeur agrégée de la borne inférieure de
l'intervalle de confiance pour une série de valeurs. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour une série de valeurs.

```
TTest1w_sig() renvoie le niveau de précision bilatéral de test t agrégé
d'étudiants pour une série de valeurs. (weight, value)
```

ttest1w_sterr

TTest1w_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour une série de valeurs.

```
TTest1w_sterr() renvoie l'erreur type agrégée de la différence moyenne de
test t d'étudiants pour une série de valeurs. (weight, value)
```

ttest1w_t

TTest1w_t() renvoie la valeur t agrégée pour une série de valeurs.

```
TTest1w_t() renvoie la valeur t agrégée pour une série de valeurs. ( weight,
value)
```

ttest1w_upper

TTest1w_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour une série de valeurs.

```
TTest1w_upper() renvoie la valeur agrégée de la borne supérieure de
l'intervalle de confiance pour une série de valeurs. (weight, value [, sig])
```

TTest_conf

TTest_conf renvoie la valeur agrégée de l'intervalle de confiance du test t pour deux échantillons indépendants.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_df

TTest_df() renvoie la valeur agrégée (degrés de liberté) du test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_dif

TTest_dif() est une fonction numérique qui renvoie la différence moyenne agrégée de test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_lower

TTest_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_sig

TTest_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_sterr

TTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_t

TTest_t() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest_t( Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest_upper

TTest_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_conf

TTestw_conf() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_df

TTestw_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_dif

TTestw_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_lower

TTestw_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_sig

TTestw_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_sterr

TTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_t

TTestw_t() renvoie la valeur t agrégée pour deux séries indépendantes de valeurs.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.

Argument	Description
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTestw_upper

TTestw_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests de deux échantillons d'étudiants indépendants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Argument	Description
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_conf

TTest1_conf() renvoie la valeur agrégée de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_df

TTest1_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_df (value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1_df( Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_dif

TTest1_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_dif (value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1_dif( Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_lower

TTest1_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_lower (value [, sig])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1_lower( value )
TTest1_lower( value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_sig

TTest1_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_sig (value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1_sig( Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_sterr

TTest1_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_sterr (value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1_sterr( Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_t

TTest1_t() renvoie la valeur t agrégée pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_t (value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1_t( Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1_upper

TTest1_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1_upper (value [, sig])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_conf

TTest1w_conf() est une fonction **numérique** qui renvoie la valeur agrégée de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1w_conf (weight, value [, sig ])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_df

TTest1w_df() renvoie la valeur df (degrés de liberté) agrégée du test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1w_df (weight, value)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1w_df( Weight, Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_dif

TTest1w_dif() renvoie la différence moyenne agrégée de test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1w_dif( Weight, Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_lower

TTest1w_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_sig

TTest1w_sig() renvoie le niveau de précision bilatéral de test t agrégé d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1w_sig( Weight, Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_sterr

TTest1w_sterr() renvoie l'erreur type agrégée de la différence moyenne de test t d'étudiants pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1w_sterr( Weight, Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_t

TTest1w_t() renvoie la valeur t agrégée pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
TTest1w_t( Weight, Value )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

TTest1w_upper

TTest1w_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique à des t-tests d'un échantillon d'étudiants où les séries de données d'entrée sont fournies dans un format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
TTest1w_upper (weight, value [, sig])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons à évaluer. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
weight	Chaque valeur définie dans l'argument **value** peut être comptée une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Voir aussi :

p *Création d'un rapport t-test type (page 523)*

Fonctions z-test

Examen statistique de deux populations moyennes. Un test z portant sur deux échantillons examine si deux échantillons sont différents. Ce test s'emploie fréquemment lorsque deux distributions normales présentent des variances connues et lorsqu'une expérience utilise une grande taille d'échantillon.

Les fonctions de test statistique z-test sont groupées d'après le type de série de données d'entrée applicable à la fonction.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Exemples d'utilisation des fonctions z-test (page 527)

Fonctions au format unicolonne

Les fonctions suivantes s'appliquent à des z-tests comportant des séries de données d'entrée simples.

ztest_conf

ZTest_conf() renvoie la valeur z agrégée pour une série de valeurs.

```
ZTest_conf() renvoie la valeur z agrégée pour une série de valeurs. (value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() renvoie la différence moyenne de test z agrégée pour une série de valeurs.

```
ZTest_dif() renvoie la différence moyenne de test z agrégée pour une série de valeurs. (value [, sigma])
```

ztest_sig

ZTest_sig() renvoie le niveau de précision bilatéral de test z agrégé pour une série de valeurs.

```
ZTest_sig() renvoie le niveau de précision bilatéral de test z agrégé pour une série de valeurs. (value [, sigma])
```

ztest_sterr

ZTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de test z pour une série de valeurs.

```
ZTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de test z pour une série de valeurs. (value [, sigma])
```

ztest_z

ZTest_z() renvoie la valeur z agrégée pour une série de valeurs.

```
ZTest_z() renvoie la valeur z agrégée pour une série de valeurs. (value [, sigma])
```

ztest_lower

ZTest_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
ZTest_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes. (grp, value [, sig [, eq_var]])
```

ztest_upper

ZTest_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
ZTest_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes. (grp, value [, sig [, eq_var]])
```

Fonctions au format bicolonne pondéré

Les fonctions suivantes s'appliquent aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

ztestw_conf

ZTestw_conf() renvoie la valeur z agrégée de l'intervalle de confiance pour une série de valeurs.

```
ZTestw_conf() renvoie la valeur z agrégée de l'intervalle de confiance pour
une série de valeurs. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif() renvoie la différence moyenne de test z agrégée pour une série de valeurs.

```
ZTestw_dif() renvoie la différence moyenne de test z agrégée pour une série
de valeurs.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
ZTestw_lower() renvoie la valeur agrégée de la borne inférieure de
l'intervalle de confiance pour deux séries de valeurs indépendantes. (weight,
value [, sigma])
```

ztestw_sig

ZTestw_sig() renvoie le niveau de précision bilatéral de test z agrégé pour une série de valeurs.

```
ZTestw_sig() renvoie le niveau de précision bilatéral de test z agrégé pour
une série de valeurs. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de test z pour une série de valeurs.

```
ZTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de
test z pour une série de valeurs. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

```
ZTestw_upper() renvoie la valeur agrégée de la borne supérieure de
l'intervalle de confiance pour deux séries de valeurs indépendantes. (weight,
value [, sigma])
```

ztestw_z

ZTestw_z() renvoie la valeur z agrégée pour une série de valeurs.

```
ZTestw_z() renvoie la valeur z agrégée pour une série de valeurs. (weight,
value [, sigma])
```

ZTest_z

ZTest_z() renvoie la valeur z agrégée pour une série de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_z(value[, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTest_z( Value-TestValue )
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_sig

ZTest_sig() renvoie le niveau de précision bilatéral de test z agrégé pour une série de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_sig(value[, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTest_sig(Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_dif

ZTest_dif() renvoie la différence moyenne de test z agrégée pour une série de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_dif(value[, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTest_dif(Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_sterr

ZTest_sterr() renvoie l'erreur type agrégée de la différence moyenne de test z pour une série de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_sterr(value[, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTest_sterr(Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_conf

ZTest_conf() renvoie la valeur z agrégée pour une série de valeurs.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_conf(value[, sigma[, sig]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTest_conf(Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_lower

ZTest_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTest_upper

ZTest_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_z

ZTestw_z() renvoie la valeur z agrégée pour une série de valeurs.

Cette fonction s'applique aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTestw_z (weight, value [, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Les valeurs doivent être renvoyées par **value**. La fonction utilise une moyenne d'échantillon de 0. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la valeur des échantillons de valeurs.
weight	Chaque échantillon de valeur défini dans l'argument **value** peut être compté une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTestw_z( Weight, Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_sig

ZTestw_sig() renvoie le niveau de précision bilatéral de test z agrégé pour une série de valeurs.

Cette fonction s'applique aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTestw_sig (weight, value [, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Les valeurs doivent être renvoyées par **value**. La fonction utilise une moyenne d'échantillon de 0. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la valeur des échantillons de valeurs.
weight	Chaque échantillon de valeur défini dans l'argument **value** peut être compté une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTestw_sig( Weight, Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_dif

ZTestw_dif() renvoie la différence moyenne de test z agrégée pour une série de valeurs.

Cette fonction s'applique aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTestw_dif ( weight, value [, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Les valeurs doivent être renvoyées par **value**. La fonction utilise une moyenne d'échantillon de 0. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la valeur des échantillons de valeurs.
weight	Chaque échantillon de valeur défini dans l'argument **value** peut être compté une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTestw_dif( Weight, Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_sterr

ZTestw_sterr() renvoie l'erreur type agrégée de la différence moyenne de test z pour une série de valeurs.

Cette fonction s'applique aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTestw_sterr (weight, value [, sigma])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Les valeurs doivent être renvoyées par **value**. La fonction utilise une moyenne d'échantillon de 0. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la valeur des échantillons de valeurs.
weight	Chaque échantillon de valeur défini dans l'argument **value** peut être compté une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTestw_sterr( Weight, Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_conf

ZTestw_conf() renvoie la valeur z agrégée de l'intervalle de confiance pour une série de valeurs.

Cette fonction s'applique aux z-tests dont la série de données d'entrée est fournie au format bicolonne pondéré.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Une population moyenne de 0 est utilisée. Si vous voulez effectuer le test sur une autre moyenne, vous devez soustraire la moyenne des échantillons de valeurs.
weight	Chaque échantillon de valeur défini dans l'argument **value** peut être compté une ou plusieurs fois en fonction d'une valeur de coefficient correspondante définie dans l'argument **weight**.
sigma	S'il est connu, l'écart type peut être défini dans l'argument **sigma**. Si **sigma** est omis, l'écart type de l'échantillon réel sera utilisé.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemple :

```
ZTestw_conf( Weight, Value-TestValue)
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_lower

ZTestw_lower() renvoie la valeur agrégée de la borne inférieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

ZTestw_upper

ZTestw_upper() renvoie la valeur agrégée de la borne supérieure de l'intervalle de confiance pour deux séries de valeurs indépendantes.

Cette fonction s'applique à des t-tests d'échantillons d'étudiants indépendants.

Si la fonction est utilisée dans le script de chargement de données, les valeurs sont itérées sur un nombre d'enregistrements définis par une clause group by.

Si la fonction est utilisée dans une expression de graphique, les valeurs sont itérées sur les dimensions du graphique.

Syntaxe :

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

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Échantillons de valeurs à évaluer. Les échantillons de valeurs doivent être groupés de manière logique en étant spécifiés par exactement deux valeurs dans l'argument **group**. Si le nom de champ des échantillons de valeurs n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Value**.
grp	Champ contenant le nom de chacun des deux échantillons de groupes. Si le nom de champ d'un groupe n'est pas spécifié dans le script de chargement, le champ se voit automatiquement attribuer le nom **Type**.
sig	Il est possible de spécifier le niveau de précision bilatéral dans **sig**. S'il est omis, **sig** est défini sur 0.025, donnant un intervalle de confiance de 95 %.
eq_var	Si l'argument **eq_var** est spécifié comme False (0), des variances distinctes sont utilisées pour les deux échantillons. Si l'argument **eq_var** est spécifié comme True (1), des variances de même valeur sont utilisées entre les échantillons.

Limitations :

Si la valeur de l'expression contient des valeurs textuelles, des valeurs NULL ou des valeurs manquantes, la fonction renvoie la valeur NULL.

Exemples :

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Voir aussi :

p *Exemples d'utilisation des fonctions z-test (page 527)*

Exemples de fonctions de test statistique

Cette section comprend des exemples de fonctions de test statistique telles qu'elles s'appliquent dans des graphiques et le script de chargement de données.

Exemples d'utilisation des fonctions chi2-test dans les graphiques

Les fonctions chi2-test permettent de déterminer les valeurs associées à une analyse statistique de type chi (ou khi) au carré.

Cette section décrit la procédure de création de visualisations à l'aide d'échantillons de données dans le but d'identifier les valeurs des fonctions de test de distribution de chi2 disponibles dans Qlik Sense. Pour une description de la syntaxe et des arguments, reportez-vous aux rubriques des différentes fonctions de graphique chi2-test.

Chargement des données des échantillons

Il y a trois ensembles d'échantillons de données décrivant trois échantillons statistiques différents à charger dans le script.

Procédez comme suit :

1. Permet de créer une nouvelle application.
2. Dans l'éditeur de chargement de données, saisissez les informations suivantes :

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the
top of the script.
Sample_1:
LOAD * inline [
Grp,Grade,Count
I,A,15
I,B,7
I,C,9
I,D,20
I,E,26
I,F,19
II,A,10
II,B,11
II,C,7
II,D,15
II,E,21
II,F,16
];
// Sample_2 data is pre-aggregated: If raw data is used, it must be aggregated using
count()...
Sample_2:
LOAD * inline [
Sex,Opinion,OpCount
1,2,58
1,1,11
1,0,10
2,2,35
2,1,25
2,0,23 ] (delimiter is ',');
// Sample_3a data is transformed using the crosstable statement...
Sample_3a:
crosstable(Gender, Actual) LOAD
Description,
[Men (Actual)] as Men,
[Women (Actual)] as Women;
LOAD * inline [
Men (Actual),Women (Actual),Description
58,35,Agree
11,25,Neutral
10,23,Disagree ] (delimiter is ',');
// Sample_3b data is transformed using the crosstable statement...
Sample_3b:
```

```
crosstable(Gender, Expected) LOAD
Description,
[Men (Expected)] as Men,
[Women (Expected)] as Women;
LOAD * inline [
Men (Expected),Women (Expected),Description
45.35,47.65,Agree
17.56,18.44,Neutral
16.09,16.91,Disagree ] (delimiter is ',');
// Sample_3a and Sample_3b will result in a (fairly harmless) Synthetic Key...
```

3. Cliquez sur l'icône pour charger les données.

Création de visualisations de fonctions de graphique chi2-test

Exemple : Échantillon 1

Procédez comme suit :

1. Dans l'éditeur de chargement de données, cliquez sur pour accéder à la vue de l'application, puis cliquez sur la feuille que vous avez créée auparavant.
   Le mode feuille s'ouvre.
2. Cliquez sur **Éditer la feuille** pour éditer la feuille.
3. Sous **Graphiques**, ajoutez une table, puis sous **Champs**, ajoutez Grp, Grade et Count en tant que dimensions.
   Cette table affiche les échantillons de données.
4. Ajoutez une autre table comportant l'expression suivante en tant que dimension :
   `ValueList('p','df','Chi2')`
   De cette manière, la fonction de dimensions synthétiques sert à créer les étiquettes des dimensions en utilisant les noms des trois fonctions chi2-test.
5. Ajoutez à la table l'expression suivante en tant que mesure :
   ```
   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)))
   ```
   Cela a pour effet de placer la valeur résultante de chaque fonction chi2-test dans la table, à côté de la dimension synthétique associée.
6. Définissez l'option **Formatage des nombres** de la mesure sur **Nombre** et choisissez **3 Chiffres significatifs**.

Dans l'expression de la mesure, vous pourriez tout aussi bien opter pour l'expression : `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Résultat :

La table résultante pour les fonctions chi2-test se rapportant aux données de l'échantillon 1 contient les valeurs suivantes :

Table des résultats

p	df	Chi2
0.820	5	2.21

Exemple : Échantillon 2

Procédez comme suit :

1. Sur la feuille que vous éditiez dans l'exemple de l'échantillon 1, sous **Graphiques**, ajoutez une table, puis sous **Champs**, ajoutez Sex, Opinion et OpCount en tant que dimensions.
2. Effectuez une copie de la table des résultats de l'échantillon 1 à l'aide des commandes **Copier** et **Coller**. Éditez l'expression de la mesure et remplacez les arguments dans les trois fonctions chi2-test par les noms des champs utilisés dans les données de l'échantillon 2, par exemple : `Chi2Test_p(Sex,Opinion,OpCount)`.

Résultat :

La table résultante pour les fonctions chi2-test se rapportant aux données de l'échantillon 2 contient les valeurs suivantes :

Table des résultats

p	df	Chi2
0.000309	2	16.2

Exemple : Échantillon 3

Procédez comme suit :

1. Créez deux tables supplémentaires de la même manière que dans les exemples de données des échantillons 1 et 2. Dans la table des dimensions, utilisez les champs suivants comme dimensions : Gender, Description, Actual et Expected.
2. Dans la table des résultats, servez-vous des noms des champs utilisés dans les données de l'échantillon 3, par exemple : `Chi2Test_p(Gender,Description,Actual,Expected)`.

Résultat :

La table résultante pour les fonctions chi2-test se rapportant aux données de l'échantillon 3 contient les valeurs suivantes :

Table des résultats

p	df	Chi2
0.000308	2	16.2

Exemples d'utilisation des fonctions chi2-test dans le script de chargement de données

Les fonctions chi2-test permettent de déterminer les valeurs associées à une analyse statistique de type chi (ou khi) au carré. Cette section explique comment utiliser les fonctions de test de distribution de chi2 disponibles dans Qlik Sense dans le script de chargement de données. Pour une description de la syntaxe et des arguments, reportez-vous aux rubriques des différentes fonctions de script chi2-test.

Cet exemple utilise une table contenant le nombre d'étudiants obtenant une note (allant de A à F) pour deux groupes de personnes (soit I et II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Chargement des échantillons de données

Procédez comme suit :

1. Permet de créer une nouvelle application.
2. Dans l'éditeur de chargement de données, saisissez les informations suivantes :
```
// 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. Cliquez sur l'icône pour charger les données.

Vous avez terminé de charger les échantillons de données.

Chargement des valeurs de fonction chi2-test

Nous allons à présent chargé dans une nouvelle table les valeurs chi2-test d'après les échantillons de données, en les groupant par Grp.

Procédez comme suit :

1. Dans l'éditeur de chargement de données, ajoutez ce qui suit à la fin du script :
```
// 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. Cliquez sur l'icône pour charger les données.

Vous avez maintenant chargé les valeurs chi2-test dans une table intitulée Chi2_table.

Résultats

Vous pouvez afficher les valeurs chi2-test résultantes dans le visionneur de modèle de données sous **Aperçu**. Elles devraient avoir l'aspect suivant :

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Création d'un rapport t-test type

Un rapport t-test type sur les étudiants peut comprendre des tables présentant les résultats des statistiques de groupes **Group Statistics** et des échantillons indépendants **Independent Samples Test**.

Au cours des sections suivantes, nous verrons comment créer ces tables à l'aide des fonctions t-test de Qlik Sense appliquées à deux groupes indépendants d'échantillons, à savoir Observation et Comparison. Les tables correspondantes pour ces échantillons auraient l'aspect suivant :

Statistiques de groupe

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

Test d'échantillon indépendant

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Chargement des échantillons de données

Procédez comme suit :

1. Créez une nouvelle application comprenant une nouvelle feuille, puis ouvrez cette dernière.
2. Saisissez les lignes suivantes dans l'éditeur de chargement de données :
```
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 '|');
```
Dans ce script de chargement, **recno()** est inclus, car **crosstable** requiert trois arguments. C'est pourquoi **recno()** fournit simplement un argument supplémentaire, dans ce cas un ID par ligne. Sans cela, les échantillons de valeurs **Comparison** ne seraient pas chargés.
3. Cliquez sur l'icône pour charger les données.

Création de la table Group Statistics

Procédez comme suit :

1. Dans l'éditeur de chargement de données, cliquez sur pour accéder à la vue de l'application, puis cliquez sur la feuille que vous avez créée auparavant.
   Le mode feuille s'ouvre.
2. Cliquez sur **Éditer la feuille** pour éditer la feuille.
3. Sous **Graphiques**, ajoutez une table et sous **Champs**, ajoutez les expressions suivantes en tant que mesures :

   Exemples d'expression

Étiquette	Expression
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Ajoutez Type comme dimension à la table.
5. Cliquez sur **Tri**, puis déplacez l'entrée Type en haut de la liste de tri.

   Résultat :

   Une table Group Statistics relative à ces échantillons aurait l'aspect suivant :

   Statistiques de groupe

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Création de la table Two Independent Sample Student's T-test

Procédez comme suit :

1. Cliquez sur **Éditer la feuille** pour éditer la feuille.
2. Ajoutez l'expression suivante comme dimension à la table. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))`

3. Sous **Graphiques**, ajoutez une table comportant les expressions suivantes en tant que mesures :

Exemples d'expression

Étiquette	Expression
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval of the Difference (Lower)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_lower(Type, Value,(1-(95)/100)/2),TTest_lower (Type, Value,(1-(95)/100)/2, 0))
95% Confidence Interval of the Difference (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper (Type, Value,(1-(95)/100)/2, 0))

Résultat :

Test d'échantillon indépendant

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Exemples d'utilisation des fonctions z-test

Les fonctions z-test permettent de déterminer les valeurs associées à une analyse statistique de type z-test pour les grands échantillons de données, généralement comptant plus de 30 éléments et pour lesquels la variance est connue.

Cette section décrit la procédure de création de visualisations à l'aide d'échantillons de données dans le but d'identifier les valeurs des fonctions z-test disponibles dans Qlik Sense. Pour une description de la syntaxe et des arguments, reportez-vous aux rubriques des différentes fonctions de graphique z-test.

Chargement des échantillons de données

Les échantillons de données utilisés dans cet exemple sont identiques à ceux des exemples de la fonction t-test. La taille des échantillons de données serait normalement considérée comme trop petite pour une analyse de type z-test, mais elle est suffisante à des fins d'illustration de l'emploi des différentes fonctions z-test dans Qlik Sense.

Procédez comme suit :

1. Créez une nouvelle application comprenant une nouvelle feuille, puis ouvrez cette dernière.

   Si vous avez créé une application pour les fonctions t-test, vous pouvez vous en servir afin de définir une nouvelle feuille pour ces fonctions.

2. Dans l'éditeur de chargement de données, saisissez les informations suivantes :

```
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 '|');
```

   Dans ce script de chargement, **recno()** est inclus, car **crosstable** requiert trois arguments. C'est pourquoi **recno()** fournit simplement un argument supplémentaire, dans ce cas un ID par ligne. Sans cela, les échantillons de valeurs **Comparison** ne seraient pas chargés.

3. Cliquez sur l'icône pour charger les données.

Création de visualisations de fonctions de graphique z-test

Procédez comme suit :

1. Dans l'éditeur de chargement de données, cliquez sur pour accéder à la vue de l'application, puis cliquez sur la feuille que vous avez créée au moment du chargement du script.
   Le mode feuille s'ouvre.
2. Cliquez sur **Éditer la feuille** pour éditer la feuille.
3. Sous **Graphiques**, ajoutez une table, puis sous **Champs**, ajoutez Type en tant que dimension.
4. Ajoutez à la table les expressions suivantes en tant que mesures.

Exemples d'expressions

Étiquette	Expression
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)

*Il peut s'avérer nécessaire d'ajuster le formatage des nombres des mesures afin d'afficher des valeurs qui ont du sens. La lisibilité de la table sera améliorée si vous définissez le formatage des nombres de la plupart des mesures sur **Nombre > Simple** au lieu du paramètre **Auto**. Cependant, pour ZTest Sig, par exemple, utilisez le formatage des nombres : **Personnalisé (es)**, puis adaptez le modèle de format à **# ##**.*

Résultat :

La table résultante pour les fonctions z-test se rapportant à l'échantillon de données contient les valeurs suivantes :

Table des résultats

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

Création de visualisations de fonctions de graphique z-testw

Les fonctions z-testw s'utilisent lorsque la série de données d'entrée est fournie au format bicolonne pondéré. Les expressions nécessitent une valeur pour l'argument weight. Dans notre cas, la valeur 2 est utilisée tout au long des exemples, mais vous pourriez très bien choisir une expression, qui définirait alors une valeur d'argument weight pour chaque observation.

Exemples et résultats :

En utilisant les mêmes échantillons de données et le même formatage des nombres que pour les fonctions z-test, la table résultante pour les fonctions z-testw contient les valeurs suivantes :

Table des résultats

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

Fonctions d'agrégation de chaînes

Cette section décrit les fonctions d'agrégation relatives aux chaînes.

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Fonctions d'agrégation de chaînes utilisées dans le script de chargement de données

Concat

Concat() permet de combiner des valeurs de chaîne. La fonction de script renvoie la concaténation de chaînes agrégée de toutes les valeurs incluses dans l'expression itérée sur un nombre donné d'enregistrements définis par une clause **group by**.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() renvoie la valeur qui a été chargée en premier à partir des enregistrements définis par l'expression et triés par une clause **group by**.

Cette fonction est uniquement disponible comme fonction de script.

```
FirstValue  (expression)
```

LastValue

LastValue() renvoie la valeur qui a été chargée en dernier à partir des enregistrements définis par l'expression et triés par une clause **group by**.

Cette fonction est uniquement disponible comme fonction de script.

```
LastValue  (expression)
```

MaxString

MaxString() recherche des valeurs de chaîne dans l'expression et renvoie la dernière valeur textuelle triée dans l'ordre alphabétique sur un nombre donné d'enregistrements, comme défini par une clause **group by**.

```
MaxString (expression )
```

MinString

MinString() recherche des valeurs de chaîne dans l'expression et renvoie la première valeur textuelle triée dans l'ordre alphabétique sur un nombre donné d'enregistrements, comme défini par une clause **group by**.

```
MinString (expression )
```

Fonctions d'agrégation de chaînes utilisées dans les graphiques

Les fonctions de graphique suivantes sont disponibles pour l'agrégation de chaînes dans les graphiques.

Concat

Concat() permet de combiner des valeurs de chaîne. Cette fonction renvoie la concaténation de chaînes agrégée de toutes les valeurs incluses dans l'expression évaluée pour chaque dimension.

```
Concat - fonction de graphique({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]] string[, delimiter[, sort_weight]])
```

MaxString

MaxString() recherche des valeurs de chaîne dans l'expression ou le champ et renvoie la dernière valeur textuelle dans l'ordre de tri alphabétique.

```
MaxString - fonction de graphique({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() recherche des valeurs de chaîne dans l'expression ou le champ et renvoie la première valeur textuelle dans l'ordre de tri alphabétique.

```
MinString - fonction de graphique({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat() permet de combiner des valeurs de chaîne. La fonction de script renvoie la concaténation de chaînes agrégée de toutes les valeurs incluses dans l'expression itérée sur un nombre donné d'enregistrements définis par une clause **group by**.

Syntaxe :

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Type de données renvoyé : chaîne

Arguments :

Expression ou champ contenant la chaîne à traiter.

Arguments

Argument	Description
string	Expression ou champ contenant la chaîne à traiter.
delimiter	Les valeurs peuvent être séparées par la chaîne indiquée dans l'argument delimiter.
sort-weight	L'ordre de concaténation peut être déterminé par la valeur de la dimension **sort-weight**, le cas échéant, avec la chaîne correspondant à la valeur la plus basse apparaissant en premier dans la concaténation..
distinct	Si le terme **distinct** précède l'expression, tous les doublons sont ignorés.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Exemples et résultats

Exemple	Résultat	Résultats après ajout à une feuille

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
`LOAD SalesGroup,Concat(Team) as TeamConcat1`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat1
		AlphaBetaDeltaGammaGamma
		EpsilonEtaThetaZeta

Supposons que la table **TeamData** est chargée comme dans l'exemple précédent :
 	
`LOAD SalesGroup,Concat(distinct Team,'-')`
`as TeamConcat2 Resident TeamData Group By`
`SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat2
		Alpha-Beta-Delta-Gamma
		Epsilon-Eta-Theta-Zeta

Exemple	Résultat	Résultats après ajout à une feuille
Supposons que la table **TeamData** est chargée comme dans l'exemple précédent. Comme l'argument de **sort-weight** est ajouté, les résultats sont triés d'après la valeur de la dimension Amount : `LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat - fonction de graphique

Concat() permet de combiner des valeurs de chaîne. Cette fonction renvoie la concaténation de chaînes agrégée de toutes les valeurs incluses dans l'expression évaluée pour chaque dimension.

Syntaxe :

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
string	Expression ou champ contenant la chaîne à traiter.
delimiter	Les valeurs peuvent être séparées par la chaîne indiquée dans l'argument delimiter.
sort-weight	L'ordre de concaténation peut être déterminé par la valeur de la dimension **sort-weight**, le cas échéant, avec la chaîne correspondant à la valeur la plus basse apparaissant en premier dans la concaténation..
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si le terme **DISTINCT** précède les arguments de la fonction, les doublons résultant de l'évaluation des arguments de la fonction sont ignorés.

Argument	Description
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Results table

SalesGroup	Amount	Concat(Team)	Concat(TOTAL <SalesGroup> Team)
East	25000	Alpha	AlphaBetaDeltaGammaGamma
East	20000	BetaGammaGamma	AlphaBetaDeltaGammaGamma
East	14000	Delta	AlphaBetaDeltaGammaGamma
West	17000	Epsilon	EpsilonEtaThetaZeta
West	14000	Eta	EpsilonEtaThetaZeta
West	23000	Theta	EpsilonEtaThetaZeta
West	19000	Zeta	EpsilonEtaThetaZeta

Exemples de fonction

Exemple	Résultat
`Concat(Team)`	La table est créée à partir des dimensions SalesGroup et Amount, et des variations de la mesure Concat(Team). Sans tenir compte du résultat des totaux, notez que même si des données existent pour huit valeurs de la colonne Team réparties sur deux valeurs de la colonne SalesGroup, le seul résultat de la mesure Concat(Team) qui aboutit à une concaténation de plus d'une valeur de chaîne Team dans la table est la ligne contenant la dimension Amount 20000, qui aboutit au résultat BetaGammaGamma. Ceci s'explique par le fait qu'il existe trois valeurs de données d'entrée pour la cellule 20000 sous Amount. Les autres résultats ne sont pas concaténés lorsque la mesure s'étend sur les dimensions, car il n'y a qu'une seule valeur Team pour chaque combinaison de SalesGroup et Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma, puisque le qualificateur DISTINCT signifie que le résultat Gamma en double est ignoré. De plus, l'argument « delimiter » est défini comme une virgule suivie d'un espace.

Exemple	Résultat
`Concat (TOTAL <SalesGroup> Team)`	Toutes les valeurs de chaîne se rapportant aux valeurs de la mesure Team sont concaténées si le qualificateur TOTAL est utilisé. Si vous spécifiez la sélection de champ <SalesGroup>, les résultats sont divisés selon les deux valeurs de la dimension SalesGroup. Pour la valeur SalesGroupEast, les résultats correspondent à AlphaBetaDeltaGammaGamma. Pour la valeur SalesGroupWest, les résultats correspondent à EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	En ajoutant l'argument pour **sort-weight** : Amount, les résultats sont triés d'après la valeur de la dimension Amount. Les résultats obtenus sont DeltaBetaGammaGammaAlpha et EtaEpsilonZEtaTheta.

Données utilisées dans l'exemple :

```
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

FirstValue() renvoie la valeur qui a été chargée en premier à partir des enregistrements définis par l'expression et triés par une clause **group by**.

Cette fonction est uniquement disponible comme fonction de script.

Syntaxe :

```
FirstValue ( expr)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur textuelle, elle renvoie la valeur NULL.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat	Résultats sur une feuille

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`FirstValue1:`
`LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		FirstTeamLoaded
		Gamma
		Zeta

LastValue

LastValue() renvoie la valeur qui a été chargée en dernier à partir des enregistrements définis par l'expression et triés par une clause **group by**.

Cette fonction est uniquement disponible comme fonction de script.

Syntaxe :

```
LastValue ( expr )
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur textuelle, elle renvoie la valeur NULL.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de notre application afin de visualiser le résultat.

Pour obtenir le même aspect que dans la colonne des résultats ci-dessous, désélectionnez le tri par ordre numérique et alphabétique. Pour ce faire, dans le panneau des propriétés, sous Tri, passez du paramètre Auto au paramètre Personnalisé(es).

Exemple	Résultat	Résultat avec tri personnalisé

`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

MaxString() recherche des valeurs de chaîne dans l'expression et renvoie la dernière valeur textuelle triée dans l'ordre alphabétique sur un nombre donné d'enregistrements, comme défini par une clause **group by**.

Syntaxe :

```
MaxString ( expr )
```

Type de données renvoyé : double

Arguments :

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur textuelle, elle renvoie la valeur NULL.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Exemple	Résultat	

`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

Supposons que la table **TeamData** est chargée comme dans l'exemple précédent et que le script de chargement de données comporte l'instruction 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 - fonction de graphique

MaxString() recherche des valeurs de chaîne dans l'expression ou le champ et renvoie la dernière valeur textuelle dans l'ordre de tri alphabétique.

Syntaxe :

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

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Limitations :

Si l'expression ne contient aucune valeur comportant de représentation de chaîne, la valeur NULL est renvoyée.

Exemples et résultats :

Table des résultats

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

Exemples de fonction

Exemple	Résultat
`MaxString (Team)`	Il existe trois valeurs de 20000 pour la dimension Amount : deux de Gamma (à des dates différentes) et une de Beta. Le résultat de la mesure MaxString (Team) correspond donc à Gamma, car il s'agit de la valeur la plus élevée dans les chaînes triées.
`MaxString (Date)`	2013/11/01 désigne la plus grande valeur de Date des trois dates associées à la dimension Amount. Cela suppose que le script contient l'instruction SET `SET DateFormat='YYYY-MM-DD';'`

Données utilisées dans l'exemple :

```
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

MinString() recherche des valeurs de chaîne dans l'expression et renvoie la première valeur textuelle triée dans l'ordre alphabétique sur un nombre donné d'enregistrements, comme défini par une clause **group by**.

Syntaxe :

```
MinString ( expr )
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur textuelle, elle renvoie la valeur NULL.

Exemples et résultats :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Données résultantes

Exemple	Résultat	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MinString(Team) as MinString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString1 Alpha Epsilon
Supposons que la table **TeamData** est chargée comme dans l'exemple précédent et que le script de chargement de données comporte l'instruction SET : `SET DateFormat='DD/MM/YYYY';`: LOAD SalesGroup,MinString(Date) as MinString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString2 01/05/2013 01/06/2013

MinString - fonction de graphique

MinString() recherche des valeurs de chaîne dans l'expression ou le champ et renvoie la première valeur textuelle dans l'ordre de tri alphabétique.

Syntaxe :

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

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
TOTAL	Si le terme **TOTAL** précède les arguments de la fonction, le calcul est effectué à partir de toutes les valeurs possibles au vu des sélections actives, et pas seulement à partir de celles qui sont associées à la valeur dimensionnelle active. Autrement dit, les dimensions du graphique ne sont pas prises en compte. En utilisant **TOTAL [<fld {.fld}>]**, où le qualificateur **TOTAL** est suivi d'un ou de plusieurs noms constituant un sous-ensemble des variables de dimension du graphique, vous créez un sous-ensemble du nombre total de valeurs possibles.

Exemples et résultats :

Échantillons de données

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

Exemples de fonction

Exemples	Résultats
`MinString (Team)`	Il existe trois valeurs de 20000 pour la dimension Amount : deux de Gamma (à des dates différentes) et une de Beta. Le résultat de la mesure MinString (Team) correspond donc à Beta, car il s'agit de la première valeur dans les chaînes triées.
`MinString (Date)`	2013/11/01 désigne la valeur de Date la plus ancienne des trois dates associées à la dimension Amount. Cela suppose que le script contient l'instruction SET `SET DateFormat='YYYY-MM-DD';'`

Données utilisées dans l'exemple :

```
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 '|');
```

Fonctions de dimension synthétique

Dans l'application, une dimension synthétique est créée à partir des valeurs générées par les fonctions de dimension synthétique et pas directement à partir des champs du modèle de données. Lorsque des valeurs générées par une fonction de dimension synthétique sont utilisées dans un graphique en tant que dimension calculée, le résultat est une dimension synthétique. Les dimensions synthétiques vous permettent de créer, par exemple, des graphiques dont les valeurs de dimensions sont issues de vos données, autrement de dimensions dynamiques.

Les dimensions synthétiques ne sont pas affectées par les sélections.

Les fonctions de dimension synthétique suivantes peuvent s'utiliser dans les graphiques.

ValueList

ValueList() renvoie un ensemble de valeurs listées qui, lorsqu'elles sont utilisées dans une dimension calculée, forment une dimension synthétique.

```
ValueList - fonction de graphique (v1 {, Expression})
```

ValueLoop

ValueLoop() renvoie un ensemble de valeurs itérées qui, lorsqu'elles sont utilisées dans une dimension calculée, forment une dimension synthétique.

```
ValueLoop - fonction de graphique(from [, to [, step ]])
```

ValueList - fonction de graphique

ValueList() renvoie un ensemble de valeurs listées qui, lorsqu'elles sont utilisées dans une dimension calculée, forment une dimension synthétique.

*Dans les graphiques comprenant une dimension synthétique créée à l'aide de la fonction **ValueList**, il est possible de faire référence à la valeur de dimension correspondant à une cellule d'expression donnée en réexécutant la fonction **ValueList** avec les mêmes paramètres dans l'expression du graphique. La fonction peut naturellement être employée n'importe où dans la disposition, mais sauf quand elle est utilisée pour des dimensions synthétiques, elle ne présente de l'intérêt que lorsqu'elle figure dans une fonction d'agrégation.*

Les dimensions synthétiques ne sont pas affectées par les sélections.

Syntaxe :

```
ValueList(v1 {,...})
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
v1	Valeur statique (généralement une chaîne, mais un nombre est également possible).
{,...}	Liste facultative de valeurs statiques.

Exemples et résultats :

Exemples de fonction

Exemple	Résultat
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Lorsque vous utilisez cette fonction pour créer une dimension dans une table, par exemple, les trois valeurs de chaîne sont converties en étiquettes de ligne dans la table. Celles-ci peuvent ensuite être référencées dans une expression.
`=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) ))`	Cette expression utilise les valeurs issues de la dimension créée et les référence dans une instruction IF imbriquée comme données d'entrée pour trois fonctions d'agrégation : **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

Données utilisées dans les exemples :

```
SalesPeople:
LOAD * INLINE [
SaleID|SalesPerson|Amount|Year
1|1|12|2013
2|1|23|2013
3|1|17|2013
4|2|9|2013
5|2|14|2013
6|2|29|2013
```

```
7|2|4|2013
8|1|15|2012
9|1|16|2012
10|2|11|2012
11|2|17|2012
12|2|7|2012
] (delimiter is '|');
```

ValueLoop - fonction de graphique

ValueLoop() renvoie un ensemble de valeurs itérées qui, lorsqu'elles sont utilisées dans une dimension calculée, forment une dimension synthétique.
La liste de valeurs générées débute par la valeur **from** et se termine par la valeur **to** et comprend des valeurs intermédiaires par incréments du pas.

*Dans les graphiques comprenant une dimension synthétique créée à l'aide de la fonction **ValueLoop**, il est possible de faire référence à la valeur de dimension correspondant à une cellule d'expression donnée en réexécutant la fonction **ValueLoop** avec les mêmes paramètres dans l'expression du graphique. La fonction peut naturellement être employée n'importe où dans la disposition, mais sauf quand elle est utilisée pour des dimensions synthétiques, elle ne présente de l'intérêt que lorsqu'elle figure dans une fonction d'agrégation.*

Les dimensions synthétiques ne sont pas affectées par les sélections.

Syntaxe :

```
ValueLoop(from [, to [, step ]])
```

Type de données renvoyé : double

Arguments :

Arguments

Arguments	Description
from	Valeur de départ de l'ensemble de valeurs à générer.
to	Valeur de fin de l'ensemble de valeurs à générer.
step	Taille de l'incrément séparant les valeurs.

Exemples et résultats :

Exemples de fonction

Exemple	Résultat
`ValueLoop (1, 10)`	Cette fonction crée une dimension dans une table qui peut, par exemple, servir à différentes fins comme l'étiquetage numéroté. L'exemple présenté ici aboutit à des valeurs numérotées de 1 à 10. Ces valeurs peuvent ensuite être référencées dans une expression.
`ValueLoop (2, 10,2)`	Cet exemple aboutit aux valeurs numérotées 2, 4, 6, 8 et 10, car l'argument step est doté d'une valeur de 2.

Agrégations imbriquées

Certaines situations peuvent nécessiter l'application d'une agrégation au résultat d'une autre agrégation. On parle alors d'agrégations imbriquées.

Dans la plupart des expressions de graphique, vous ne pouvez pas imbriquer d'agrégations. En revanche, vous pouvez imbriquer des agrégations si vous utilisez le qualificateur **TOTAL** dans la fonction d'agrégation interne.

Il est interdit d'utiliser plus de 100 niveaux d'imbrication.

Agrégations imbriquées avec utilisation du qualificateur TOTAL

Exemple :

Supposons que vous souhaitiez calculer la somme du champ **Sales** mais inclure uniquement les transactions dont la date de commande **OrderDate** correspond à l'année dernière. L'année passée peut être obtenue via la fonction d'agrégation **`Max(TOTAL`** `Year(OrderDate)`**`)`**.

L'agrégation suivante renverrait le résultat souhaité :

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Pour ce type d'imbrication, Qlik Sense nécessite l'inclusion du qualificateur **TOTAL**. Il est nécessaire pour obtenir la comparaison souhaitée. Ce type d'emploi de l'imbrication est relativement courant et conseillé.

Voir aussi :

p *Aggr - fonction de graphique (page 545)*

5.3 Aggr - fonction de graphique

Aggr() renvoie un tableau de valeurs pour l'expression calculée au moyen de la ou des dimensions définies. Par exemple, la valeur maximale de ventes, par client et par région.

La fonction **Aggr** est utilisée pour les agrégations imbriquées dans lesquelles le première paramètre (l'agrégation interne) est calculé une fois par valeur dimensionnelle. Les dimensions sont spécifiées dans le deuxième paramètre (et les paramètres suivants).

De plus, la fonction **Aggr** doit être imbriquée dans une fonction d'agrégation externe en utilisant le tableau de résultats provenant de la fonction **Aggr** comme entrée de l'agrégation dans laquelle elle est imbriquée.

Syntaxe :

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

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression composée d'une fonction d'agrégation. Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection.
StructuredParameter	StructuredParameter se compose d'une dimension et éventuellement de critères de tri au format suivant : `(Dimension(Sort-type, Ordering))` La dimension est un champ unique, qui ne peut pas correspondre à une expression. La dimension permet de déterminer le tableau de valeurs pour lequel l'expression Aggr est calculée. Si des critères de tri sont inclus, le tableau de valeurs créé par la fonction Aggr, calculé pour la dimension, est trié. Ceci est important lorsque l'ordre de tri affecte le résultat de l'expression dans laquelle la fonction Aggr est incluse. Pour en savoir plus sur l'utilisation des critères de tri, reportez-vous à la rubrique Ajout de critères de tri à la dimension dans le paramètre structuré.
SetExpression	Par défaut, la fonction d'agrégation couvre l'ensemble des enregistrements possibles définis par la sélection. Il est possible de définir un ensemble d'enregistrements alternatif à l'aide d'une expression d'analyse d'ensembles.
DISTINCT	Si l'argument expression est précédé du qualificateur **distinct** ou si aucun qualificateur n'est utilisé, chaque combinaison de valeurs de dimension génère une seule valeur de retour. Il s'agit du mode de création standard des agrégations ; chaque combinaison distincte de valeurs de dimension correspond à une ligne dans le graphique.
NODISTINCT	Si l'argument expression est précédé du qualificateur **nodistinct**, chaque combinaison de valeurs de dimension peut, suivant la structure de données sous-jacente, générer plusieurs valeurs de retour. En présence d'une seule dimension, la fonction **aggr** renvoie un tableau comprenant autant d'éléments que les données de la source comportent de lignes.

Les fonctions d'agrégation de base (telles que **Sum**, **Min** et **Avg**) renvoient une valeur numérique unique, tandis que la fonction Aggr() est comparable à la création d'un ensemble de résultats graduel temporaire (table virtuelle), sur lequel une autre agrégation peut être effectuée. Par exemple, lorsque vous calculez une valeur de vente moyenne en additionnant les ventes par client dans une instruction **Aggr()**, puis en calculant la moyenne des résultats additionnés : **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Si vous souhaitez créer des agrégations de graphiques imbriqués à plusieurs niveaux, utilisez la fonction Aggr() dans des dimensions calculées.

Limitations :

Dans une fonction Aggr(), chaque dimension doit consister en un seul champ et être différente d'une expression (dimension calculée).

Ajout de critères de tri à la dimension dans le paramètre structuré

Dans sa forme de base, l'argument StructuredParameter dans la syntaxe de la fonction Aggr est une dimension unique. L'expression Aggr(Sum(Sales, Month)) détermine la valeur totale des ventes pour chaque mois. Toutefois, lorsqu'elle est incluse dans une autre fonction d'agrégation, des résultats inattendus peuvent survenir si aucun critère de tri n'est utilisé. En effet, certaines dimensions peuvent être triées par ordre numérique ou alphabétique, et ainsi de suite.

Dans l'argument StructuredParameter de la fonction Aggr, vous pouvez indiquer des critères de tri sur la dimension dans votre expression. De cette manière, vous imposez un ordre de tri sur la table virtuelle qui est créée par la fonction Aggr.

L'argument StructuredParameter présente la syntaxe suivante :

```
(FieldName, (Sort-type, Ordering))
```

Les paramètres structurés peuvent être imbriqués :

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Les types de tri possibles sont les suivants : NUMERIC, TEXT, FREQUENCY ou LOAD_ORDER.

Les types de classement associés à chaque type de tri sont les suivants :

Types de classement autorisés

Type de tri	Types de classement autorisés
NUMERIC	ASCENDING, DESCENDING ou REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE ou Z2A
FREQUENCY	DESCENDING, REVERSE ou ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING ou REVERSE

Les types de classement REVERSE et DESCENDING sont équivalents.

Pour le type de tri TEXT, les types de classement ASCENDING et A2Z sont équivalents, et DESCENDING, REVERSE et Z2A sont équivalents.

Pour le type de tri LOAD_ORDER, les types de classement ASCENDING et ORIGINAL sont équivalents.

Exemples : Expressions de graphique via Aggr

Exemples - expressions de graphique

Exemple d'expression de graphique 1

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple d'expression de graphique ci-dessous.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|CC|2|20 Betacab|DD|25|25
Canutility|AA|8|15 Canutility|CC|0|19 ] (delimiter is '|');
```

Expression de graphique

Créez une visualisation d'indicateur KPI dans une feuille Qlik Sense. Ajoutez l'expression suivante à l'indicateur KPI, sous forme de mesure :

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Résultat

376.7

Explication

L'expression `Aggr(Sum(UnitSales*UnitPrice), Customer)` détermine la valeur totale des ventes par **Customer** et renvoie un tableau de valeurs : 295, 715 et 120 pour les trois valeurs **Customer**.

De fait, nous avons établi une liste temporaire de valeurs sans avoir à créer de table ou colonne explicite contenant ces valeurs.

Ces valeurs sont utilisées comme données d'entrée pour la fonction **Avg()** afin de rechercher la valeur moyenne des ventes, soit 376.7.

Exemple d'expression de graphique 2

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple d'expression de graphique ci-dessous.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|BB|7|12
Betacab|CC|2|22 Betacab|CC|4|20 Betacab|DD|25|25 Canutility|AA|8|15 Canutility|AA|5|11
Canutility|CC|0|19 ] (delimiter is '|');
```

Expression de graphique

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Customer**, **Product**, **UnitPrice** et **UnitSales**. Ajoutez à la table l'expression suivante, sous forme de mesure :

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Résultat

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Explication

Liste de valeurs : 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 et 19. Le qualificateur **nodistinct** signifie que le tableau contient un élément pour chaque ligne des données source : chacun d'eux correspond à la valeur **UnitPrice** maximale pour chaque cellule **Customer** et chaque cellule **Product**.

Exemple d'expression de graphique 3

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple d'expression de graphique ci-dessous.

```
Set vNumberOfOrders = 1000;  OrderLines: Load   RowNo() as OrderLineID,         OrderID,        OrderDate,
Round((Year(OrderDate)-2005)*1000*Rand()*Rand()*Rand1) as Sales         While Rand()<=0.5 or IterNo
()=1; Load * Where OrderDate<=Today(); Load     Rand() as Rand1,        Date(MakeDate(2013)+Floor
((365*4+1)*Rand())) as OrderDate,       RecNo() as OrderID      Autogenerate vNumberOfOrders;
Calendar: Load distinct         Year(OrderDate) as Year,        Month(OrderDate) as Month,      OrderDate
Resident OrderLines;
```

Expressions de graphique

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Year** et **Month**. Ajoutez à la table les expressions suivantes sous forme de mesures :

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` libellé comme Structured Aggr() dans la table.

Résultat

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075
2013	Mar	25651	127726
2013	Apr	36585	164311
2013	May	61211	225522
2013	Jun	23689	249211
2013	Jul	42311	291522
2013	Aug	41913	333435
2013	Sep	28886	362361
2013	Oct	25977	388298
2013	Nov	44455	432753
2013	Dec	64144	496897
2014	Jan	67775	67775

Explication

Cet exemple affiche les valeurs agrégées sur une période de douze mois pour chaque année dans l'ordre chronologique croissant, d'où la partie (Numeric, Ascending) des paramètres structurés de l'expression **Aggr()**. Deux dimensions spécifiques sont requises comme paramètres structurés : **Year** et **Month**, (1) **Year** (valeur numérique) et (2) **Month** (valeur numérique) triées. Ces deux dimensions doivent être utilisées dans la visualisation de table ou de graphique. Cela est nécessaire pour que la liste de dimensions de la fonction **Aggr()** corresponde aux dimensions de l'objet utilisé dans la visualisation.

Vous pouvez comparer la différence entre ces mesures dans une table ou dans des graphiques en courbes distincts :

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Vous devez voir clairement que seule la dernière expression effectue l'accumulation de valeurs agrégées souhaitée.

Voir aussi :

p *Fonctions d'agrégation de base (page 327)*

5.4 Fonctions de couleur

Ces fonctions de graphique peuvent s'utiliser dans des expressions associées à la définition et à l'évaluation des propriétés de couleur des objets de graphique, de même que dans les scripts de chargement de données.

Pour des raisons de comptabilité avec les versions antérieures, Qlik Sense prend en charge les fonctions de couleur ***Color()****,* ***qliktechblue*** *et* ***qliktechgray****, mais leur utilisation est déconseillée.*

ARGB

ARGB() est utilisée dans des expressions afin de spécifier ou d'évaluer les propriétés de couleur d'un objet graphique, où la couleur est définie par des composants rouge **r**, vert **g** et bleu **b**, ainsi qu'un facteur alpha (d'opacité) **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() est utilisée dans des expressions afin de spécifier ou d'évaluer les propriétés de couleur d'un objet graphique, où la couleur est définie par une valeur de teinte **hue**, une valeur de **saturation** et une valeur de luminosité **luminosity** comprise entre 0 et 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() renvoie un entier correspondant au code de couleur de la couleur définie par les trois paramètres : le composant rouge r, le composant vert g (pour green) et le composant bleu b. Ces composants doivent avoir des valeurs entières comprises entre 0 et 255. La fonction peut être utilisée dans des expressions pour définir ou évaluer les propriétés de couleur d'un objet de graphique.

```
RGB  (r, g, b)
```

Colormix1

La fonction **Colormix1()** s'utilise dans les expressions pour renvoyer une représentation ARGB à partir d'un dégradé de deux couleurs, basé sur une valeur comprise entre 0 et 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value est un nombre réel compris entre 0 et 1.

- Si Value = 0 ColorZero est renvoyé.
- Si Value = 1 ColorOne est renvoyé.
- Si 0 < Value < 1, l'ombrage intermédiaire approprié est renvoyé.

ColorZero est une représentation RGB valide correspondant à l'extrémité inférieure de l'intervalle.

ColorOne est une représentation RGB valide correspondant à l'extrémité supérieure de l'intervalle.

Exemple :

```
Colormix1(0.5, red(), blue())
```

renvoie :

```
ARGB(255,64,0,64) (purple)
```

Colormix2

La fonction **Colormix2()** est utilisée dans les expressions pour renvoyer une représentation ARGB à partir d'un dégradé de deux couleurs, sur la base d'une valeur comprise entre -1 et 1, avec la possibilité de spécifier une couleur intermédiaire pour la position centrale (0).

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value est un nombre réel compris entre -1 et 1.

- Si Value = -1, la première couleur est renvoyée.
- Si Value = 1, la deuxième couleur est renvoyée.
- Si -1 < Value < 1, le mélange de couleur approprié est renvoyé.

ColorMinusOne est une représentation RGB valide correspondant à l'extrémité inférieure de l'intervalle.

ColorOne est une représentation RGB valide correspondant à l'extrémité supérieure de l'intervalle.

ColorZero est une représentation RGB valide facultative correspondant au centre de l'intervalle.

SysColor

SysColor() renvoie la représentation ARGB de la couleur système Windows nr, où nr correspond au paramètre de la fonction API de Windows **GetSysColor(nr)**.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() renvoie une valeur ARGB d'une couleur provenant d'une table de couleurs faisant varier le composant de teinte du modèle de couleurs HSV. La table de couleurs commence par le rouge, passe par le jaune, le vert, le cyan, le bleu, le magenta, puis revient au rouge. x doit être spécifié sous forme de valeur comprise entre 0 et 1.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() renvoie une valeur ARGB d'une couleur provenant d'une table de couleurs commençant par le bleu, passant par le cyan, le jaune et l'orange, puis revenant au rouge. x doit être spécifié sous forme de valeur comprise entre 0 et 1.

```
ColorMapJet (x)
```

Fonctions de couleur prédéfinies

Les fonctions suivantes peuvent être utilisées dans les expressions pour les couleurs prédéfinies. Chaque fonction renvoie une représentation RGB.

Vous avez la possibilité de fournir un paramètre facultatif pour le facteur alpha, auquel cas une représentation de couleurs ARGB est renvoyée. Un facteur alpha égal à 0 correspond à une transparence totale tandis qu'un facteur alpha égal à 255 traduit une opacité totale. Si aucune valeur n'est saisie pour alpha, elle est supposée correspondre à 255.

Fonctions de couleur prédéfinies

Fonction de couleur	RGB Valeur
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)

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() est utilisée dans des expressions afin de spécifier ou d'évaluer les propriétés de couleur d'un objet graphique, où la couleur est définie par des composants rouge **r**, vert **g** et bleu **b**, ainsi qu'un facteur alpha (d'opacité) **alpha**.

Syntaxe :

```
ARGB(alpha, r, g, b)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
alpha	Valeur de transparence comprise entre 0 et 255. 0 correspond à une transparence totale et 255 à une opacité complète.
r, g, b	Valeurs des composants rouge, vert et bleu. Un composant de couleur 0 ne correspond à aucune contribution tandis qu'un composant de couleur 255 signifie une contribution complète.

Tous les arguments doivent être des expressions dont le résultat est un entier compris entre 0 et 255.

Si vous interprétez et formatez le composant numérique selon la notation hexadécimale, les valeurs des composants de couleur sont plus faciles à identifier. Par exemple, le vert clair est identifié par le nombre 4 278 255 360, ce qui donne FF00FF00 en notation hexadécimale. Les deux premières positions 'FF' (255) représentent le canal **alpha**. Les deux positions '00' suivantes représentent la quantité de **rouge**, les deux positions 'FF' suivantes représentent la quantité de **vert** et les deux dernières positions '00' représentent la quantité de **bleu**..

RGB

RGB() renvoie un entier correspondant au code de couleur de la couleur définie par les trois paramètres : le composant rouge r, le composant vert g (pour green) et le composant bleu b. Ces composants doivent avoir des valeurs entières comprises entre 0 et 255. La fonction peut être utilisée dans des expressions

pour définir ou évaluer les propriétés de couleur d'un objet de graphique.

Syntaxe :

```
RGB (r, g, b)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
r, g, b	Valeurs des composants rouge, vert et bleu. Un composant de couleur 0 ne correspond à aucune contribution tandis qu'un composant de couleur 255 signifie une contribution complète.

Tous les arguments doivent être des expressions dont le résultat est un entier compris entre 0 et 255.

Si vous interprétez et formatez le composant numérique selon la notation hexadécimale, les valeurs des composants de couleur sont plus faciles à identifier. Par exemple, le vert clair est identifié par le nombre 4 278 255 360, ce qui donne FF00FF00 en notation hexadécimale. Les deux premières positions 'FF' (255) représentent le canal **alpha**. Dans les fonctions **RGB** et **HSL**, la valeur correspond toujours à FF (opaque). Les deux positions '00' suivantes représentent la quantité de **rouge**, les deux positions 'FF' suivantes représentent la quantité de **vert** et les deux dernières positions '00' représentent la quantité de **bleu**..

Exemple : Expression de graphique
Cet exemple applique une couleur personnalisée à un graphique :

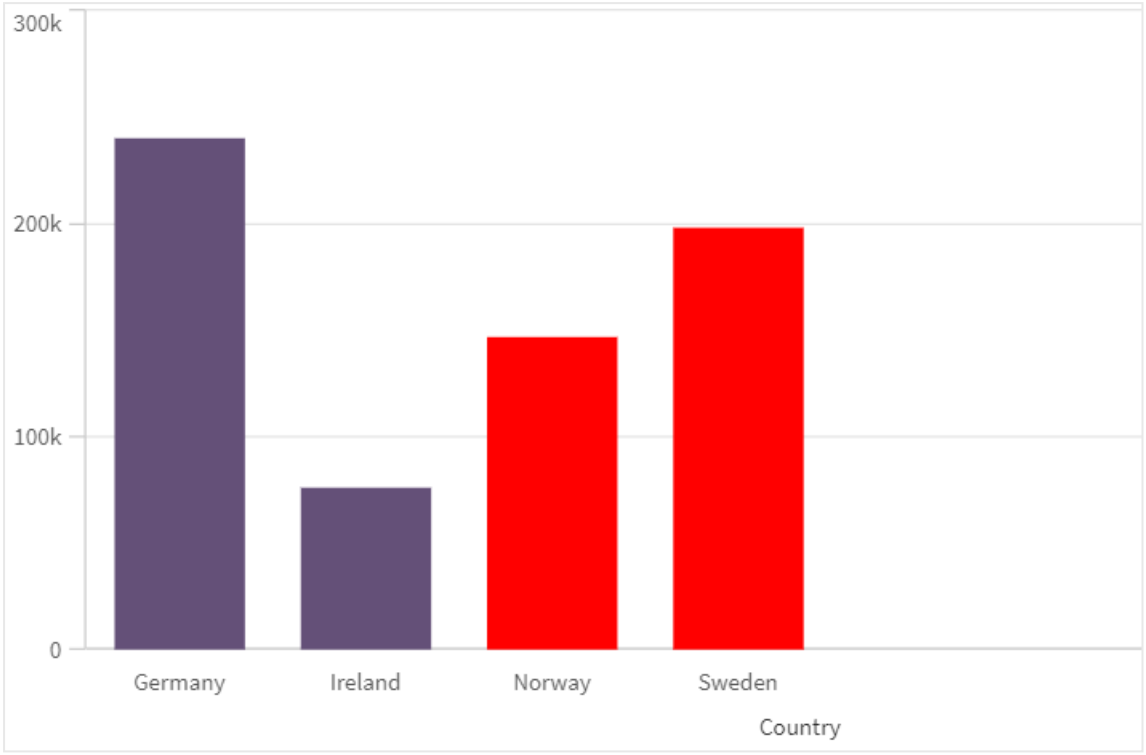
Données utilisées dans cet exemple :

```
ProductSales: Load * Inline [Country,Sales,Budget Sweden,100000,50000 Germany, 125000, 175000 Norway, 74850, 68500 Ireland, 45000, 48000 Sweden,98000,50000 Germany, 115000, 175000 Norway, 71850, 68500 Ireland, 31000, 48000 ] (delimiter is ',');
```

Saisissez l'expression suivante dans le panneau des propriétés **Couleurs et légende** :

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Résultat :



Exemple : Script de chargement

L'exemple suivant affiche les valeurs RVB équivalentes aux valeurs au format hexagonal :

```
Load Text(R & G & B) as Text, RGB(R,G,B)      as Color; Load Num#(R,'(HEX)') as R, Num#(G,'(HEX)') as G, Num#(B,'(HEX)') as B Inline [R,G,B 01,02,03 AA,BB,CC];
```

Résultat :

Texte	Couleur
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() est utilisée dans des expressions afin de spécifier ou d'évaluer les propriétés de couleur d'un objet graphique, où la couleur est définie par une valeur de teinte **hue**, une valeur de **saturation** et une valeur de luminosité **luminosity** comprise entre 0 et 1.

Syntaxe :

```
HSL (hue, saturation, luminosity)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
hue, saturation, luminosity	Valeurs des composants de teinte, saturation et luminosité (hue, saturation et luminosity) comprises entre 0 et 1.

Tous les arguments doivent être des expressions dont le résultat est un entier compris entre 0 et 1.

Si vous interprétez et formatez le composant numérique selon la notation hexadécimale, les valeurs RGB des composants de couleur sont plus faciles à identifier. Par exemple, le vert clair est identifié par le nombre 4 278 255 360, ce qui donne FF00FF00 et RGB (0,255,0) en notation hexadécimale. Cela équivaut à HSL (80/240, 240/240, 120/240) - une valeur HSL de (0.33, 1, 0.5).

5.5 Fonctions conditionnelles

Les fonctions conditionnelles évaluent toutes une condition avant de renvoyer différentes réponses suivant la valeur de la condition. Les fonctions s'utilisent aussi bien dans le script de chargement de données que dans les expressions de graphique.

Vue d'ensemble des fonctions conditionnelles

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

alt

La fonction **alt** renvoie le premier des paramètres doté d'une représentation numérique valide. Si aucune correspondance n'est trouvée, c'est le dernier paramètre qui est renvoyé. Vous pouvez utiliser autant de paramètres que vous le souhaitez.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

La fonction **class** attribue le premier paramètre à un intervalle de classe. Le résultat est une valeur double utilisant a<=x<b comme valeur textuelle, où a et b sont les limites supérieure et inférieure de la série, et la limite inférieure comme valeur numérique.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

La fonction **coalesce** renvoie le premier des paramètres doté d'une représentation non-NULL valide. Vous pouvez utiliser autant de paramètres que vous le souhaitez.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

La fonction **if** renvoie une valeur variant selon que la condition fournie avec la fonction est évaluée comme True ou False.

```
if (condition , then , else)
```

match

La fonction **match** compare le premier paramètre à tous les paramètres suivants et renvoie l'emplacement numérique des expressions qui correspondent. La comparaison est sensible à la casse des caractères.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

La fonction **mixmatch** compare le premier paramètre à tous les paramètres suivants et renvoie l'emplacement numérique des expressions qui correspondent. La comparaison n'est pas sensible à la casse des caractères.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

La fonction pick renvoie la *n*ième expression de la liste.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

La fonction **wildmatch** compare le premier paramètre à tous les paramètres suivants et renvoie le numéro de l'expression qui correspond. Elle permet d'utiliser des caractères génériques (***** et **?**) dans les chaînes de comparaison. ***** correspond à n'importe quelle séquence de caractères. **?** correspond à n'importe quel caractère unique. La comparaison n'est pas sensible à la casse des caractères.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

La fonction **alt** renvoie le premier des paramètres doté d'une représentation numérique valide. Si aucune correspondance n'est trouvée, c'est le dernier paramètre qui est renvoyé. Vous pouvez utiliser autant de paramètres que vous le souhaitez.

Syntaxe :

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Arguments :

Arguments

Argument	Description
expr1	Première expression utilisée pour rechercher une représentation numérique valide.
expr2	Deuxième expression utilisée pour rechercher une représentation numérique valide.
expr3	Troisième expression utilisée pour rechercher une représentation numérique valide.
else	Valeur à renvoyer si aucun des paramètres précédents ne comporte de représentation numérique valide.

La fonction alt s'utilise souvent avec les fonctions d'interprétation des nombres et des dates. De cette manière, Qlik Sense peut tester différents formats de date selon un ordre de priorité. Elle peut également servir à gérer les valeurs NULL dans les expressions numériques.

Exemples :

Exemples

Exemple	Résultat

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	Cette expression recherche dans le champ date une date correspondant à l'un des trois formats de date spécifiés. Si elle en trouve une, elle renvoie une valeur double comportant la chaîne d'origine et une représentation numérique valide d'une date. Si aucune correspondance n'est trouvée, elle renvoie le texte 'No valid date' (sans représentation numérique valide).

`alt(Sales,0) + alt(Margin,0)`	Cette expression ajoute les champs Sales et Margin, remplaçant les valeurs manquantes (NULL) par un 0.

class

La fonction **class** attribue le premier paramètre à un intervalle de classe. Le résultat est une valeur double utilisant a<=x<b comme valeur textuelle, où a et b sont les limites supérieure et inférieure de la série, et la limite inférieure comme valeur numérique.

Syntaxe :

```
class(expression, interval [ , label [ , offset ]])
```

Arguments :

Arguments

Argument	Description
interval	Nombre spécifiant la taille de la série.

Argument	Description
label	Chaîne arbitraire pouvant remplacer le caractère x dans le texte du résultat.
offset	Nombre pouvant être utilisé comme décalage par rapport au point de départ par défaut de la classification. Le point de départ par défaut est généralement égal à 0.

Exemples :

Exemples

Exemple	Résultat
`class( var,10 )` avec `var = 23`	renvoie '20<=x<30'
`class( var,5,'value' )` avec `var = 23`	renvoie '20<= value <25'
`class( var,10,'x',5 )` avec `var = 23`	renvoie '15<=x<25'

Exemple - Script de chargement avec class

Exemple : script de chargement

Script de chargement

Dans cet exemple, nous chargeons une table contenant le nom et l'âge de plusieurs personnes. Nous voulons ajouter un champ qui classe chaque personne selon un groupe d'âges couvrant un intervalle de dix ans. La table source d'origine ressemble à ce qui suit.

Résultats

Name	Age
John	25
Karen	42
Yoshi	53

Pour ajouter le champ de classification par groupe d'âges, vous pouvez insérer une instruction load antérieure à l'aide de la fonction **class**.

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Créez le tableau ci-dessous dans Qlik Sense pour afficher les résultats.

```
 LOAD *, class(Age, 10, 'age') As Agegroup;  LOAD * INLINE [ Age, Name 25, John 42, Karen 53, Yoshi];
```

Résultats

Résultats

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

La fonction **coalesce** renvoie le premier des paramètres doté d'une représentation non-NULL valide. Vous pouvez utiliser autant de paramètres que vous le souhaitez.

Syntaxe :

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Arguments :

Arguments

Argument	Description
expr1	Première expression utilisée pour rechercher une représentation non NULL valide.
expr2	Deuxième expression utilisée pour rechercher une représentation non NULL valide.
expr3	Troisième expression utilisée pour rechercher une représentation non NULL valide.

Exemples :

Exemples

Exemple	Résultat
	Cette expression transforme l'ensemble des valeurs NULL d'un champ en 'N/A'.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Cette expression effectue une sélection entre trois différents champs de description de produit lorsque certains champs n'ont pas de valeurs pour le produit. Le premier des champs, dans l'ordre donné, avec une valeur non NULL est renvoyé. Si aucun des champs ne contient de valeur, le résultat est 'Aucune description disponible'.

Exemple	Résultat
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Cette expression coupe les guillemets englobants potentiels du champ *FileName*. Si la valeur *FileName* donnée est entre guillemets, ceux-ci sont supprimés et la valeur *FileName* sans guillemets incluse est renvoyée. Si la fonction *TextBetween* ne trouve pas les délimiteurs, elle renvoie une valeur nulle, que **Coalesce** rejette, renvoyant à la place la valeur *FileName* brute.

if

La fonction **if** renvoie une valeur variant selon que la condition fournie avec la fonction est évaluée comme True ou False.

Syntaxe :

```
if(condition , then [, else])
```

Arguments

Argument	Description
condition	Expression interprétée de manière logique.
then	Expression pouvant être de tout type. Si la *condition* est True, la fonction if renvoie la valeur de l'expression *then*.
else	Expression pouvant être de tout type. Si la *condition* est False, la fonction if renvoie la valeur de l'expression *else*. Ce paramètre est facultatif. Si la *condition* est False, NULL est renvoyé si vous n'avez pas précisé else.

Exemple

Exemple	Résultat
`if( Amount>= 0, 'OK', 'Alarm' )`	Cette expression teste si la valeur est un nombre positif (0 ou une valeur supérieure) et renvoie 'OK' si c'est le cas. Si le résultat est inférieur à 0, l'expression renvoie 'Alarm'.

Exemple - Script de chargement avec if

Exemple : Script de chargement

Script de chargement

Vous pouvez utiliser If dans un script de chargement avec d'autres méthodes et objets, y compris des variables. Par exemple, si vous définissez une variable *threshold* et si vous souhaitez inclure dans le modèle de données un champ basé sur ce seuil, procédez comme suit.

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Créez le tableau ci-dessous dans Qlik Sense pour afficher les résultats.

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

set threshold = 100;

/* Create new table called Transaction_Buckets
Compare transaction_amount field from Transaction table to threshold of 100.
Output results into a new field called Compared to Threshold
 */

Transaction_Buckets:
Load
   transaction_id,
   If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)')
as [Compared to Threshold]
Resident Transactions;
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *if* dans le script de chargement.

transaction_id	Par rapport au seuil
3750	Inférieur à 100
3751	Supérieur à 100
3752	Inférieur à 100
3753	Supérieur à 100
3754	Supérieur à 100
3756	Inférieur à 100
3757	Supérieur à 100

Exemples - Expressions de graphique avec if

Exemples : Expressions de graphique

Expression de graphique 1

Script de chargement

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Après avoir chargé les données, créez les exemples d'expression de graphique ci-dessous dans un tableau Qlik Sense.

```
MyTable:
LOAD * inline [Date, Location, Incidents
1/3/2016, Beijing, 0
1/3/2016, Boston, 12
1/3/2016, Stockholm, 3
1/3/2016, Toronto, 0
1/4/2016, Beijing, 0
1/4/2016, Boston, 8];
```

Table Qlik Sense affichant des exemples de la fonction *if* utilisée dans une expression de graphique.

Date	Emplacement	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Beijing	0	Ok	Ok
1/3/2016	Boston	12	Critical	Critical
1/3/2016	Stockholm	3	Ok	Warning
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Beijing	0	Ok	Ok
1/4/2016	Boston	8	Ok	Warning

Expression de graphique 2

Dans une nouvelle application, ajoutez le script suivant dans un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données. Vous pouvez ensuite créer la table à l'aide des expressions de graphique ci-dessous.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Table Qlik Sense affichant un exemple de la fonction *if* utilisée dans une expression de graphique.

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sat	WeekEnd
1/2/2022	Sun	WeekEnd
1/3/2022	Mon	Normal Day
1/4/2022	Tue	Normal Day
1/5/2022	Wed	Normal Day
1/6/2022	Thu	Normal Day
1/7/2022	Fri	Normal Day
1/8/2022	Sat	WeekEnd
1/9/2022	Sun	WeekEnd
1/10/2022	Mon	Normal Day
1/11/2022	Tue	Normal Day
1/12/2022	Wed	Normal Day
1/13/2022	Thu	Normal Day
1/14/2022	Fri	Normal Day

match

La fonction **match** compare le premier paramètre à tous les paramètres suivants et renvoie l'emplacement numérique des expressions qui correspondent. La comparaison est sensible à la casse des caractères.

Syntaxe :

```
match( str, expr1 [ , expr2,...exprN ])
```

*Si vous souhaitez établir des comparaisons sans tenir compte de la casse, utilisez la fonction **mixmatch**. Si vous souhaitez établir des comparaisons sans tenir compte de la casse avec des caractères génériques, utilisez la fonction **wildmatch**.*

Exemple : Script de chargement avec match

Exemple : Script de chargement

Script de chargement

La fonction match permet de charger un sous-ensemble de données. Par exemple, vous pouvez renvoyer une valeur numérique pour une expression dans la fonction. Vous pouvez ensuite limiter les données chargées d'après la valeur numérique. Match renvoie 0 en l'absence de correspondance. Dans cet exemple, toutes les expressions sans correspondance renverront donc 0 et seront exclues du chargement de données par l'instruction WHERE.

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Créez le tableau ci-dessous dans Qlik Sense pour afficher les résultats.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753,
20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756,
20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /*
Create new table called Transaction_Buckets Create new fields called Customer, and Color code
- Blue and Black Load Transactions table. Match returns 1 for 'Blue', 2 for 'Black'.  Does not
return a value for 'blue' because match is case sensitive. Only values that returned numeric
value greater than 0  are loaded by WHERE statment into Transactions_Buckets table. */
Transaction_Buckets: Load  customer_id,  customer_id as [Customer],   color_code as [Color
Code Blue and Black] Resident Transactions Where match(color_code,'Blue','Black') > 0;
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction match dans le script de chargement

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Exemples - Expressions de graphique avec match

Exemples : Expressions de graphique

Expression de graphique 1

Script de chargement

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Après avoir chargé les données, créez les exemples d'expression de graphique ci-dessous dans un tableau Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

La première expression de la table ci-dessous renvoie 0 pour Stockholm, car « Stockholm » ne figure pas dans la liste d'expressions de la fonction **match**. Elle renvoie également 0 pour « Zurich », car la comparaison **match** respecte la casse des caractères.

Table Qlik Sense affichant des exemples de la fonction *match* utilisée dans une expression de graphique

Cities	match(Cities,'Toronto','Boston','Beijing','Zurich')	match(Cities,'Toronto','Boston','Beijing','Stockholm','zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	0	5

Expression de graphique 2

La fonction match vous permet de définir un tri personnalisé pour une expression.

Par défaut, les colonnes sont triées par nombre ou par ordre alphabétique, selon les données.

Table Qlik Sense affichant un exemple de l'ordre de tri par défaut

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Pour modifier l'ordre, procédez comme suit :

1. Ouvrez la section **Tri** relative au graphique dans le **panneau des propriétés**.
2. Désactivez le tri automatique défini pour la colonne à laquelle vous souhaitez appliquer un tri personnalisé.
3. Désélectionnez **Trier par nombre** et **Trier par ordre alphabétique**.
4. Sélectionnez **Trier par expression**, puis saisissez une expression semblable à la suivante :
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   L'ordre de tri appliqué à la colonne Cities est modifié en conséquence.

Table Qlik Sense affichant un exemple de modification de l'ordre de tri à l'aide de la fonction *match*

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Vous pouvez également afficher la valeur numérique renvoyée.

Table Qlik Sense affichant un exemple des valeurs numériques renvoyées par la fonction *match*

Cities	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

La fonction **mixmatch** compare le premier paramètre à tous les paramètres suivants et renvoie l'emplacement numérique des expressions qui correspondent. La comparaison n'est pas sensible à la casse des caractères.

Syntaxe :

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Si vous souhaitez plutôt établir des comparaisons tenant compte de la casse, utilisez la fonction **match**. Si vous souhaitez établir des comparaisons sans tenir compte de la casse avec des caractères génériques, utilisez la fonction **wildmatch**.

Exemple - Script de chargement avec mixmatch

Exemple : Script de chargement

Script de chargement

La fonction mixmatch permet de charger un sous-ensemble de données. Par exemple, vous pouvez renvoyer une valeur numérique pour une expression dans la fonction. Vous pouvez ensuite limiter les données chargées d'après la valeur numérique. Mixmatch renvoie 0 en l'absence de correspondance. Dans cet exemple, toutes les expressions sans correspondance renverront donc 0 et seront exclues du chargement de données par l'instruction WHERE.

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Créez le tableau ci-dessous dans Qlik Sense pour afficher les résultats.

```
Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity,
customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907,
556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00,
7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2,
2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];  /* Create new table called
Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue Load
Transactions table. Mixmatch returns 1 for 'Black', 2 for 'Blue'.  Also returns 3 for 'blue'
because mixmatch is not case sensitive. Only values that returned numeric value greater than 0
 are loaded by WHERE statement into Transactions_Buckets table. */  Transaction_Buckets: Load
  customer_id,     customer_id as [Customer],    color_code as [Color Code - Black, Blue,
blue] Resident Transactions Where mixmatch(color_code,'Black','Blue') > 0;
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction mixmatch dans le script de chargement.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
Blue	5646471

Exemples - Expressions de graphique avec mixmatch

Exemples : Expressions de graphique

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Après avoir chargé les données, créez les exemples d'expression de graphique ci-dessous dans un tableau Qlik Sense.

Expression de graphique 1

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

La première expression de la table ci-dessous renvoie 0 pour Stockholm, car « Stockholm » ne figure pas dans la liste d'expressions de la fonction **mixmatch**. Elle renvoie 4 pour « Zurich », car la comparaison **mixmatch** ne respecte pas la casse des caractères.

Table Qlik Sense affichant des exemples de la fonction *mixmatch* utilisée dans une expression de graphique

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Expression de graphique 2

Vous pouvez utiliser mixmatch pour effectuer un tri personnalisé pour une expression.

Par défaut, les colonnes sont triées par nombre ou par ordre alphabétique, selon les données.

Table Qlik Sense affichant un exemple de l'ordre de tri par défaut

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Pour modifier l'ordre, procédez comme suit :

1. Ouvrez la section **Tri** relative au graphique dans le **panneau des propriétés**.
2. Désactivez le tri automatique défini pour la colonne à laquelle vous souhaitez appliquer un tri personnalisé.
3. Désélectionnez **Trier par nombre** et **Trier par ordre alphabétique**.
4. Sélectionnez **Trier par expression**, puis saisissez l'expression suivante :

```
=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')
```

L'ordre de tri appliqué à la colonne Cities est modifié en conséquence.

Table Qlik Sense affichant un exemple de modification de l'ordre de tri à l'aide de la fonction *mixmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Vous pouvez également afficher la valeur numérique renvoyée.

Table Qlik Sense affichant un exemple des valeurs numériques renvoyées par la fonction *mixmatch*.

Cities	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

La fonction pick renvoie la *n*ième expression de la liste.

Syntaxe :

```
pick(n, expr1[ , expr2,...exprN])
```

Arguments :

Arguments

Argument	Description
n	*n* est un entier compris entre 1 et N.

Exemple :

Exemple

Exemple	Résultat
`pick( N, 'A','B',4, 6 )`	renvoie 'B' si N = 2. renvoie 4 si N = 3.

wildmatch

La fonction **wildmatch** compare le premier paramètre à tous les paramètres suivants et renvoie le numéro de l'expression qui correspond. Elle permet d'utiliser des caractères génériques (* et ?) dans les chaînes de comparaison. * correspond à n'importe quelle séquence de caractères. ? correspond à n'importe quel caractère unique. La comparaison n'est pas sensible à la casse des caractères.

Syntaxe :

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Si vous souhaitez établir des comparaisons sans tenir compte des caractères génériques, utilisez les fonctions **match** ou **mixmatch**.

Exemple : Script de chargement avec wildmatch

Exemple : Script de chargement

Script de chargement

La fonction wildmatch permet de charger un sous-ensemble de données. Par exemple, vous pouvez renvoyer une valeur numérique pour une expression dans la fonction. Vous pouvez ensuite limiter les données chargées d'après la valeur numérique. Wildmatch renvoie 0 en l'absence de correspondance. Dans cet exemple, toutes les expressions sans correspondance renverront donc 0 et seront exclues du chargement de données par l'instruction WHERE.

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Créez le tableau ci-dessous dans Qlik Sense pour afficher les résultats.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753,
20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756,
20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /*
Create new table called Transaction_Buckets Create new fields called Customer, and Color code
- Black, Blue, blue, red Load Transactions table. Wildmatch returns 1 for 'Black', 'Blue', and
'blue', and 2 for 'Red'.  Only values that returned numeric value greater than 0  are loaded
by WHERE statement into Transactions_Buckets table. */   Transaction_Buckets: Load
customer_id,    customer_id as [Customer],    color_code as [Color Code Black, Blue, blue,
Red] Resident Transactions Where wildmatch(color_code,'Bl*','R??') > 0;
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *wildmatch* dans le script de chargement

Color Code Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blue	2038593
Blue	5646471
Red	049681
Red	2038593

Exemples : Expressions de graphique avec wildmatch

Exemple : Expression de graphique

Expression de graphique 1

Créez un nouvel onglet dans l'éditeur de chargement de données, puis chargez les données suivantes sous forme de chargement inline. Après avoir chargé les données, créez les exemples d'expression de graphique ci-dessous dans un tableau Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

La première expression de la table ci-dessous renvoie 0 pour Stockholm, car « Stockholm » ne figure pas dans la liste d'expressions de la fonction **wildmatch**. Elle renvoie également 0 pour « Boston », car ? correspond uniquement à un caractère unique.

Table Qlik Sense affichant des exemples de la fonction *wildmatch* utilisée dans une expression de graphique

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Expression de graphique 2

Vous pouvez utiliser wildmatch pour effectuer un tri personnalisé pour une expression.

Par défaut, les colonnes sont triées par nombre ou par ordre alphabétique, selon les données.

Table Qlik Sense affichant un exemple de l'ordre de tri par défaut

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Pour modifier l'ordre, procédez comme suit :

1. Ouvrez la section **Tri** relative au graphique dans le **panneau des propriétés**.
2. Désactivez le tri automatique défini pour la colonne à laquelle vous souhaitez appliquer un tri personnalisé.
3. Désélectionnez **Trier par nombre** et **Trier par ordre alphabétique**.
4. Sélectionnez **Trier par expression**, puis saisissez une expression semblable à la suivante :
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   L'ordre de tri appliqué à la colonne Cities est modifié en conséquence.

Table Qlik Sense affichant un exemple de modification de l'ordre de tri à l'aide de la fonction *wildmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Vous pouvez également afficher la valeur numérique renvoyée.

Table Qlik Sense affichant un exemple des valeurs numériques renvoyées par la fonction *wildmatch*

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Fonctions de décompte

Cette section décrit les fonctions relatives aux décomptes d'enregistrements lors de l'évaluation de l'instruction **LOAD** dans le script de chargement de données. La seule fonction qu'il est possible d'utiliser dans les expressions de graphique est la fonction **RowNo()**.

Certaines fonctions de décompte sont dépourvues de paramètres, bien que les parenthèses de fin soient tout de même requises.

Vue d'ensemble des fonctions de décompte

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

autonumber

Cette fonction de script renvoie une valeur entière unique pour chaque valeur évaluée distincte de l'*expression* rencontrée au cours de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Cette fonction de script calcule un hachage de 128 bits des valeurs de l'expression d'entrée combinées et renvoie une valeur entière unique pour chaque valeur de hachage distincte rencontrée lors de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Cette fonction de script calcule un hachage de 256 bits des valeurs de l'expression d'entrée combinées et renvoie une valeur entière unique pour chaque valeur de hachage distincte rencontrée lors de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

```
autonumberhash256 (expression {, expression})
```

IterNo

Cette fonction de script renvoie un entier indiquant combien de temps un seul enregistrement est évalué dans une instruction **LOAD** avec une clause **while**. La première itération porte le numéro 1. La fonction **IterNo** n'est valable que si elle est utilisée avec une clause **while**.

```
IterNo ( )
```

RecNo

Cette fonction de script renvoie un entier correspondant au numéro de la ligne en cours de lecture dans la table active. Le premier enregistrement porte le numéro 1.

```
RecNo ( )
```

RowNo - script function

Cette fonction renvoie un entier pour la position de la ligne active dans la table interne Qlik Sense résultante. La première ligne porte le numéro 1.

```
RowNo ( )
```

RowNo - chart function

RowNo() renvoie le numéro de la ligne active dans le segment de colonne actif d'un tableau. Pour les graphiques bitmap, **RowNo()** renvoie le numéro de la ligne active dans l'équivalent du tableau simple du graphique.

```
RowNo - fonction de graphique([TOTAL])
```

autonumber

Cette fonction de script renvoie une valeur entière unique pour chaque valeur évaluée distincte de l'*expression* rencontrée au cours de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

> *Vous pouvez uniquement connecter des clés **autonumber** qui ont été générées dans le même chargement de données, car l'entier est généré en fonction de l'ordre de lecture de la table. Si vous devez utiliser des clés persistantes entre les chargements de données, indépendantes du tri des données source, optez pour les fonctions **hash128**, **hash160** ou **hash256**.*

Syntaxe :

```
autonumber(expression[ , AutoID])
```

Arguments :

Argument	Description
AutoID	Pour créer plusieurs instances de décompte lorsque la fonction **autonumber** est appliquée à différentes clés dans le script, vous pouvez utiliser le paramètre facultatif *AutoID* afin de nommer chaque décompte.

Exemple : Création d'une clé composée

Dans cet exemple, nous créons une clé composée en utilisant la fonction **autonumber** pour préserver la mémoire. L'exemple est court pour les besoins de la démonstration, mais il serait pertinent avec une table contenant un grand nombre de lignes.

Données d'exemple

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

Les données source sont chargées à l'aide de données intégrées. Ensuite, nous ajoutons une instruction load antérieure qui crée une clé composée à partir des champs Region, Year et Month.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

La table résultante a l'aspect suivant :

Table des résultats

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

Dans cet exemple, vous pouvez renvoyer à la clé RYMkey, pour l'exemple 1, au lieu de la chaîne 'North2014May', si vous devez créer un lien vers une autre table.

Passons maintenant au chargement d'une table source de coûts de manière similaire. Les champs Region, Year et Month sont exclus de l'instruction load antérieure afin d'éviter de créer une clé synthétique. Nous créons déjà une clé composée avec la fonction **autonumber**, ce qui lie les tables.

```
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
];
```

Nous pouvons à présent ajouter une visualisation de table à une feuille, puis ajouter les champs Region, Year et Month, ainsi que les mesures Sum pour les ventes et les coûts. La table aura l'aspect suivant :

Table des résultats

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Cette fonction de script calcule un hachage de 128 bits des valeurs de l'expression d'entrée combinées et renvoie une valeur entière unique pour chaque valeur de hachage distincte rencontrée lors de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

Vous pouvez uniquement connecter des clés ***autonumberhash128*** *qui ont été générées dans le même chargement de données, car l'entier est généré en fonction de l'ordre de lecture de la table. Si vous devez utiliser des clés persistantes entre les chargements de données, indépendantes du tri des données source, optez pour les fonctions* ***hash128****,* ***hash160*** *ou* ***hash256****.*

Syntaxe :

```
autonumberhash128(expression {, expression})
```

Exemple : Création d'une clé composée

Dans cet exemple, nous créons une clé composée en utilisant la fonction **autonumberhash128** pour préserver la mémoire. L'exemple est court pour les besoins de la démonstration, mais il serait pertinent avec une table contenant un grand nombre de lignes.

Données d'exemple

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

Les données source sont chargées à l'aide de données intégrées. Ensuite, nous ajoutons une instruction load antérieure qui crée une clé composée à partir des champs Region, Year et 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
];
```

La table résultante a l'aspect suivant :

Table des résultats

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

Dans cet exemple, vous pouvez renvoyer à la clé RYMkey, pour l'exemple 1, au lieu de la chaîne 'North2014May', si vous devez créer un lien vers une autre table.

Passons maintenant au chargement d'une table source de coûts de manière similaire. Les champs Region, Year et Month sont exclus de l'instruction load antérieure afin d'éviter de créer une clé synthétique. Nous créons déjà une clé composée avec la fonction **autonumberhash128**, ce qui lie les tables.

```
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
];
```

Nous pouvons à présent ajouter une visualisation de table à une feuille, puis ajouter les champs Region, Year et Month, ainsi que les mesures Sum pour les ventes et les coûts. La table aura l'aspect suivant :

Table des résultats

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Cette fonction de script calcule un hachage de 256 bits des valeurs de l'expression d'entrée combinées et renvoie une valeur entière unique pour chaque valeur de hachage distincte rencontrée lors de l'exécution du script. Cette fonction s'utilise, par exemple, pour créer une représentation mémoire compacte de clé complexe.

*Vous pouvez uniquement connecter des clés **autonumberhash256** qui ont été générées dans le même chargement de données, car l'entier est généré en fonction de l'ordre de lecture de la table. Si vous devez utiliser des clés persistantes entre les chargements de données, indépendantes du tri des données source, optez pour les fonctions **hash128**, **hash160** ou **hash256**.*

Syntaxe :

```
autonumberhash256(expression {, expression})
```

Exemple : Création d'une clé composée

Dans cet exemple, nous créons une clé composée en utilisant la fonction **autonumberhash256** pour préserver la mémoire. L'exemple est court pour les besoins de la démonstration, mais il serait pertinent avec une table contenant un grand nombre de lignes.

Exemple de table

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

Les données source sont chargées à l'aide de données intégrées. Ensuite, nous ajoutons une instruction load antérieure qui crée une clé composée à partir des champs Region, Year et 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
];
```

La table résultante a l'aspect suivant :

Table des résultats

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2

Region	Year	Month	Sales	RYMkey
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

Dans cet exemple, vous pouvez renvoyer à la clé RYMkey, pour l'exemple 1, au lieu de la chaîne 'North2014May', si vous devez créer un lien vers une autre table.

Passons maintenant au chargement d'une table source de coûts de manière similaire. Les champs Region, Year et Month sont exclus de l'instruction load antérieure afin d'éviter de créer une clé synthétique. Nous créons déjà une clé composée avec la fonction **autonumberhash256**, ce qui lie les tables.

```
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
];
```

Nous pouvons à présent ajouter une visualisation de table à une feuille, puis ajouter les champs Region, Year et Month, ainsi que les mesures Sum pour les ventes et les coûts. La table aura l'aspect suivant :

Table des résultats

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

Cette fonction de script renvoie un entier indiquant combien de temps un seul enregistrement est évalué dans une instruction **LOAD** avec une clause **while**. La première itération porte le numéro 1. La fonction **IterNo** n'est valable que si elle est utilisée avec une clause **while**.

Syntaxe :

```
IterNo( )
```

Exemples et résultats :

Exemple :

```
LOAD
   IterNo() as Day,
   Date( StartDate + IterNo() - 1 ) as Date
   While StartDate + IterNo() - 1 <= EndDate;

LOAD * INLINE
[StartDate, EndDate
2014-01-22, 2014-01-26
];
```

Cette instruction **LOAD** génère un enregistrement par date dans la plage définie par **StartDate** et **EndDate**.

Dans cet exemple, la table résultante est la suivante :

Table des résultats

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Cette fonction de script renvoie un entier correspondant au numéro de la ligne en cours de lecture dans la table active. Le premier enregistrement porte le numéro 1.

Syntaxe :

```
RecNo( )
```

Contrairement à **RowNo( )**, qui compte les lignes dans la table Qlik Sense résultante, **RecNo( )** compte les enregistrements dans la table de données brutes et est réinitialisé lorsqu'une table de données brutes est concaténée dans une autre.

Exemple : Script de chargement de données

Chargement de table de données brutes :

```
Tab1:
LOAD * INLINE
[A, B
```

```
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Chargement de numéros d'enregistrement et de ligne pour les lignes sélectionnées :

```
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;
```

Table interne Qlik Sense qui en résulte :

Table des résultats

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Cette fonction renvoie un entier pour la position de la ligne active dans la table interne Qlik Sense résultante. La première ligne porte le numéro 1.

Syntaxe :

```
RowNo( [TOTAL])
```

Contrairement à **RecNo( )**, qui compte les enregistrements dans la table de données brutes, la fonction **RowNo( )** ne compte pas les enregistrements exclus par les clauses **where** et elle n'est pas réinitialisée suite à la concaténation d'une table de données brutes dans une autre.

Si vous utilisez l'instruction load antérieure, c'est-à-dire plusieurs instructions ***LOAD*** *empilées effectuant une lecture dans la même table, vous ne pouvez employer la fonction* ***RowNo( )*** *que dans la première instruction* ***LOAD*** *. Si vous utilisez la fonction* ***RowNo( )*** *dans les instructions* ***LOAD*** *suivantes, la valeur 0 est renvoyée.*

Exemple : Script de chargement de données

Chargement de table de données brutes :

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Chargement de numéros d'enregistrement et de ligne pour les lignes sélectionnées :

```
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;
```

Table interne Qlik Sense qui en résulte :

Table des résultats

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - fonction de graphique

RowNo() renvoie le numéro de la ligne active dans le segment de colonne actif d'un tableau. Pour les graphiques bitmap, **RowNo()** renvoie le numéro de la ligne active dans l'équivalent du tableau simple du graphique.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Segments de colonne

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Syntaxe :

```
RowNo([TOTAL])
```

Type de données renvoyé : entier

Arguments :

Argument	Description
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Exemple : Expression de graphique via RowNo

Exemple - expression de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
```

```
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|5|4|19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Expression de graphique

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Customer** et **UnitSales**. Ajoutez `RowNo( )` et `RowNo(TOTAL)` comme mesures libellées **Row in Segment** et **Row Number**, respectivement. Ajoutez à la table l'expression suivante en tant que mesure :

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Résultat

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

Explication

La colonne **Row in Segment** affiche les résultats 1,2,3 pour le segment de colonne contenant les valeurs de UnitSales pour le client Astrida. La numérotation des lignes reprend alors à 1 pour le segment de colonne suivant, c'est-à-dire Betacab.

La colonne **Row Number** ignore les dimensions, à cause de l'argument `TOTAL` pour `RowNo()`, et compte les lignes du tableau.

Cette expression renvoie 0 pour la première ligne de chaque segment de colonne. De ce fait, la colonne affiche :

0, 2.25, 1.1111111, 0, 2.5, 5, 0, 2, 0 et 4.

Voir aussi :

p *Above - fonction de graphique (page 1277)*

5.7 Fonctions de date et heure

Les fonctions de date et heure de Qlik Sense sont destinées à transformer et à convertir les valeurs de date et heure. Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

Ces fonctions sont basées sur un numéro de série date-heure équivalant au nombre de jours qui se sont écoulés depuis le 30 décembre 1899. La valeur entière représente le jour tandis que la valeur fractionnaire correspond à l'heure du jour.

Qlik Sense utilise la valeur numérique du paramètre. De ce fait, un nombre est également valide en tant que paramètre lorsqu'il n'est pas formaté sous forme de date ou d'heure. Si le paramètre ne correspond pas à une valeur numérique, par exemple s'il s'agit d'une chaîne, Qlik Sense tente d'interpréter la chaîne en fonction des variables d'environnement de date et heure.

Si le format de l'heure utilisé dans le paramètre ne correspond pas à celui défini dans les variables d'environnement, Qlik Sense n'est pas en mesure d'effectuer une interprétation correcte. Pour résoudre ce problème, modifiez les paramètres ou utilisez une fonction d'interprétation.

Dans les exemples fournis pour chaque fonction, nous partons du principe que les formats d'heure et de date par défaut hh:mm:ss et YYYY-MM-DD (ISO 8601) sont utilisés.

Lors du traitement d'un horodatage à l'aide d'une fonction de date ou d'heure, Qlik Sense ignore tous les paramètres d'heure d'été, à moins que la fonction de date ou d'heure n'inclut une position géographique.

Par exemple, `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` utiliserait les paramètres d'heure d'été, contrairement à `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')`.

Vue d'ensemble des fonctions de date et heure

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Expressions entières de temps

second

Cette fonction renvoie un entier représentant la seconde au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

```
second (expression)
```

minute

Cette fonction renvoie un entier représentant la minute au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

```
minute (expression)
```

hour

Cette fonction renvoie un entier représentant l'heure au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

```
hour (expression)
```

day

Cette fonction renvoie un entier représentant le jour au cours duquel la fraction de l'**expression** est interprétée comme une date selon l'interprétation standard des nombres.

```
day (expression)
```

week

Cette fonction renvoie un entier représentant le numéro de la semaine selon la norme ISO 8601. Le numéro de la semaine est calculé à partir de l'interprétation de date de l'expression, conformément à l'interprétation standard des nombres.

```
week (expression)
```

month

Cette fonction renvoie une valeur double : un nom de mois tel que défini dans la variable d'environnement **MonthNames** et un entier compris entre 1 et 12. Le mois est calculé à partir de l'interprétation de date de l'expression, conformément à l'interprétation standard des nombres.

```
month (expression)
```

year

Cette fonction renvoie un entier représentant l'année au cours de laquelle l'**expression** est interprétée comme une date selon l'interprétation standard des nombres.

```
year (expression)
```

weekyear

Cette fonction renvoie l'année à laquelle le numéro de semaine appartient selon la norme ISO 8601. Le numéro de la semaine est compris entre 1 et environ 52.

```
weekyear (expression)
```

weekday

Cette fonction renvoie une valeur double avec :

- Un nom de jour tel que défini dans la variable d'environnement **DayNames**.
- Un entier compris entre 0 et 6 correspondant au jour nominal de la semaine (0-6).

```
weekday (date)
```

Fonctions d'horodatage

now

Cette fonction renvoie un horodatage de l'heure actuelle. La fonction renvoie des valeurs au format de variable système **TimeStamp**. La valeur **timer_mode** par défaut est 1.

```
now ([ timer_mode])
```

today

Cette fonction renvoie la date actuelle. La fonction renvoie des valeurs au format de variable système `DateFormat`.

```
today ([timer_mode])
```

LocalTime

Cette fonction renvoie un horodatage de l'heure actuelle pour un fuseau horaire donné.

```
localtime ([timezone [, ignoreDST ]])
```

Fonctions de calcul de date et d'heure make

makedate

Cette fonction renvoie une date calculée à partir de l'année **YYYY**, du mois **MM** et du jour **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Cette fonction renvoie une date calculée à partir de l'année **YYYY**, de la semaine **WW** et du jour de la semaine **D**.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Cette fonction renvoie une heure calculée à partir de l'heure **hh**, de la minute **mm** et de la seconde **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Autres fonctions de date

AddMonths

Cette fonction renvoie la date correspondant à **n** mois après la date **startdate** ou, si **n** est une valeur négative, la date correspondant à **n** mois avant la date **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Cette fonction renvoie la date correspondant à **n** années après la date **startdate** ou, si **n** est une valeur négative, la date correspondant à **n** années avant la date **startdate**.

```
addyears (startdate, n)
```

yeartodate

Cette fonction permet de déterminer si l'horodatage tombe dans l'année de la date à laquelle le script a été chargé pour la dernière fois et renvoie True si c'est le cas ou False si ce n'est pas le cas.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Fonctions de fuseau horaire

timezone

Cette fonction renvoie le fuseau horaire tel que défini sur l'ordinateur sur lequel le moteur Qlik est exécuté.

```
timezone ( )
```

GMT

Cette fonction renvoie l'heure Greenwich Mean Time actuelle, indiquée par les paramètres régionaux.

```
GMT ( )
```

UTC

Renvoie la valeur actuelle de l'argument Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Renvoie le réglage actif de l'heure d'été, tel qu'il est défini dans Windows.

```
daylightsaving ( )
```

converttolocaltime

Convertit un horodatage UTC ou GMT en heure locale sous la forme d'une valeur double. Il peut s'agir de n'importe quelle ville, n'importe quelle région ou n'importe quel fuseau horaire dans le monde.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Fonctions de réglage de l'heure

setdateyear

Cette fonction utilise comme données d'entrée un horodatage **timestamp** et une année **year** , puis elle met à jour l'horodatage **timestamp** avec l'année **year** spécifiée dans les données d'entrée.

```
setdateyear  (timestamp, year)
```

setdateyearmonth

Cette fonction utilise comme données d'entrée un horodatage **timestamp**, un mois **month** et une année **year** , puis elle met à jour l'horodatage **timestamp** avec l'année **year** et le mois **month** spécifiés dans les données d'entrée.

```
setdateyearmonth  (timestamp, year, month)
```

Fonctions d'inclusion in...

inyear

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans l'année comprenant l'argument **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie de l'année comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

```
inyeartodate  (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le trimestre comprenant l'argument **base_date**.

```
inquarter  (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie du trimestre comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

```
inquartertodate  (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le mois comprenant l'argument **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Renvoie la valeur True si l'argument **date** se trouve dans la partie du mois comprenant l'argument **basedate** jusqu'à la dernière milliseconde spécifiée dans **basedate**.

```
inmonthtodate  (date, basedate , shift)
```

inmonths

Cette fonction permet de déterminer si un horodatage tombe pendant la même période (mois, période de deux mois, trimestre, période de quatre mois ou semestre) que la date de référence. Il est également possible de déterminer si l'horodatage se situe dans une période passée ou future.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Cette fonction permet de déterminer si un horodatage tombe dans la partie d'une période (mois, période de deux mois, trimestre, période de quatre mois ou semestre) jusqu'à la dernière milliseconde incluse spécifiée dans `base_date`. Il est également possible de déterminer si l'horodatage se situe dans une

période passée ou future.

```
inmonthstodate  (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la semaine comprenant l'argument **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie de la semaine comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

Cette fonction détermine si l'argument **timestamp** tombe pendant la semaine lunaire comprenant l'argument **base_date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine. À l'exception de la dernière semaine de l'année, chaque semaine contiendra exactement sept jours.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

Cette fonction détermine si l'argument **timestamp** se trouve dans la partie de la semaine lunaire jusqu'à la dernière milliseconde spécifiée dans **base_date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le jour comprenant l'argument **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie du jour comprenant l'argument **base_timestamp** jusqu'à la milliseconde exacte spécifiée dans **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Fonctions de début... et de fin

yearstart

Cette fonction renvoie un horodatage correspondant au début du premier jour de l'année contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour de l'année contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Cette fonction renvoie une année composée de quatre chiffres comme valeur d'affichage avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour de l'année contenant l'argument **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du trimestre contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du trimestre contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Cette fonction renvoie une valeur d'affichage présentant les mois du trimestre (formatés selon la variable de script **MonthNames**) et l'année avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour du trimestre.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour du mois contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
monthstart (date [, shift = 0])
```

monthend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour du mois contenant l'argument `date`. Le format de sortie par défaut correspond à l'argument `DateFormat` défini dans le script.

```
monthend (date [, shift = 0])
```

monthname

Cette fonction renvoie une valeur d'affichage présentant le mois (formaté selon la variable de script **MonthNames**) et l'année avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour du mois.

```
monthname (date [, shift = 0])
```

monthsstart

Cette fonction renvoie une valeur correspondant à l'horodatage de la première milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence. Il est également possible de rechercher l'horodatage d'une période passée ou future.Le format de sortie par défaut correspond au format de date **DateFormat** défini dans le script.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence. Il est également possible de rechercher l'horodatage d'une période passée ou future.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Cette fonction renvoie une valeur d'affichage représentant la plage des mois de la période (formatée d'après la variable de script **MonthNames**) de même que l'année. La valeur numérique sous-jacente correspond à un horodatage de la première milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour de la semaine calendaire contenant l'argument **date**. Le format de sortie par défaut correspond au format de date **DateFormat** défini dans le script.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour (dimanche) de la semaine calendaire contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Cette fonction renvoie une valeur affichant l'année et le numéro de la semaine avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour de la semaine contenant l'argument **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Cette fonction renvoie une valeur d'affichage indiquant l'année et le numéro de la semaine lunaire correspondant à un horodatage de la première milliseconde du premier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du jour figurant dans l'argument **time**. Le format de sortie par défaut correspond à l'argument **TimestampFormat** défini dans le script.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du jour figurant dans **time**. Le format de sortie par défaut correspond à l'argument **TimestampFormat** défini dans le script.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Cette fonction renvoie une valeur affichant la date avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du jour contenant l'argument **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Fonctions de numérotation des jours

age

La fonction **age** renvoie l'âge atteint à l'heure définie dans l'argument **timestamp** (en années) d'une personne née le jour défini par la valeur **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays

La fonction **networkdays** renvoie le nombre de jours ouvrables (du lundi au vendredi) compris entre les valeurs **start_date** et **end_date** (incluses) en tenant compte de tous les arguments **holiday** facultatifs répertoriés.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

La fonction **firstworkdate** renvoie la date de début la plus récente pour atteindre la valeur **no_of_workdays** (du lundi au vendredi) se terminant au plus tard à la date définie par la valeur **end_date**, en tenant compte des jours de congé facultatifs indiqués. Les valeurs des arguments **end_date** et **holiday** doivent correspondre à des dates ou à des horodatages valides.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

La fonction **lastworkdate** renvoie la première date de fin permettant d'atteindre la valeur de l'argument **no_of_workdays** (du lundi au vendredi) si celle-ci commence à la date définie par **start_date** en tenant compte de tous les arguments **holiday** facultatifs répertoriés. Les valeurs des arguments **start_date** et **holiday** doivent correspondre à des dates ou à des horodatages valides.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Cette fonction calcule le numéro de jour de l'année dans lequel tombe un horodatage. Le calcul est effectué à partir de la première milliseconde du premier jour de l'année, mais le premier mois peut être décalé.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Cette fonction calcule le numéro de jour du trimestre dans lequel tombe un horodatage. Cette fonction est utilisée lors de la création d'un Calendrier principal.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Cette fonction renvoie la date correspondant à **n** mois après la date **startdate** ou, si **n** est une valeur négative, la date correspondant à **n** mois avant la date **startdate**.

Syntaxe :

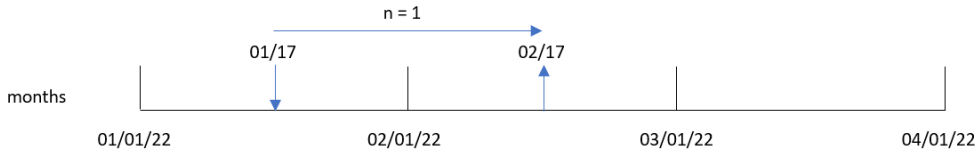
```
AddMonths(startdate, n , [ , mode])
```

Type de données renvoyé : double

La fonction `addmonths()` ajoute ou soustrait un nombre défini de mois, `n`, d'une `startdate`, et renvoie la date obtenue.

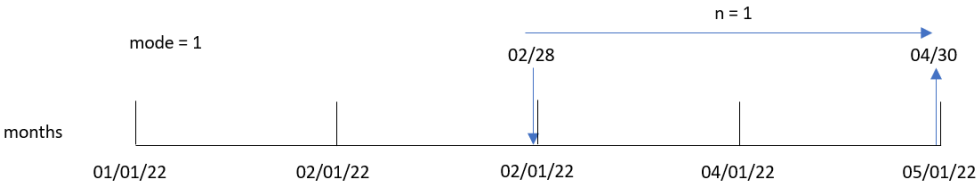
L'argument `mode` aura un impact sur les valeurs `startdate` le 28 du mois ou après. Si l'argument `mode` est défini sur 1, la fonction `addmonths()` renvoie une date égale en distance relative à la fin du mois comme `startdate`.

Exemple de diagramme de la fonction `addmonths()`



Par exemple, le 28 février est le dernier mois du mois. Si la fonction `addmonths()`, avec un `mode` égal à 1, est utilisée pour renvoyer la date deux mois plus tard, la fonction renverra la dernière date d'avril, le 30 avril.

Exemple de diagramme de la fonction `addmonths()`, avec `mode=1`



Arguments

Argument	Description
startdate	Date de début sous forme d'horodatage, par exemple '2012-10-12'.
n	Nombre de mois sous forme d'entier positif ou négatif.
mode	Indique si le mois est ajouté par rapport au début ou à la fin du mois. Le mode par défaut est 0 pour les ajouts par rapport au début du mois. Définissez le mode sur 1 pour les ajouts par rapport à la fin du mois. Lorsque le mode est défini sur 1 et que la date d'entrée est le 28 ou un jour ultérieur, la fonction vérifie le nombre de jours restants jusqu'à la fin du mois par rapport à la date de début. Le nombre de jours restants jusqu'à la fin du mois est défini dans la date renvoyée.

Cas d'utilisation

La fonction `addmonths()` sera couramment utilisée dans une expression pour trouver une date tombant un nombre de mois donné avant ou après une période.

Par exemple, la fonction `addmonths()` peut être utilisée pour identifier la date de fin de contrats de téléphone portable.

Exemples de fonction

Exemple	Résultat
`addmonths ('01/29/2003' ,3)`	Renvoie '04/29/2003'.
`addmonths ('01/29/2003',3,0)`	Renvoie '04/29/2003'.
`addmonths ('01/29/2003',3,1)`	Renvoie '04/28/2003'.
`addmonths ('01/29/2003',1,0)`	Renvoie '02/28/2003'.
`addmonths ('01/29/2003',1,1)`	Renvoie '02/26/2003'.
`addmonths ('02/28/2003',1,0)`	Renvoie '03/28/2003'.
`addmonths ('02/28/2003',1,1)`	Renvoie '03/31/2003'.
`addmonths ('01/29/2003',-3)`	Renvoie '10/29/2002'.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.

- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `two_months_later`, qui renvoie la date tombant deux mois après la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        addmonths(date,2) as two_months_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `two_months_later`

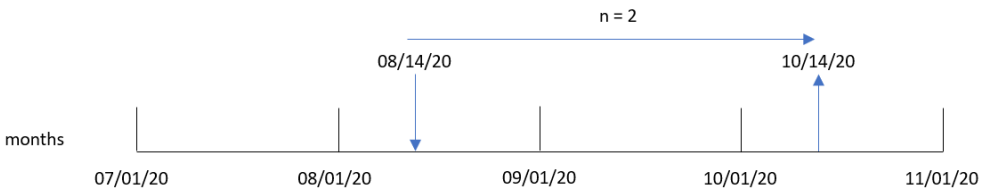
Tableau de résultats

date	two_months_later
01/10/2020	03/10/2020

date	two_months_later
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Le champ `two_months_later` est créé dans l'instruction LOAD précédente à l'aide de la fonction `addmonths()`. Le premier argument fourni identifie la date en cours d'évaluation. Le deuxième argument est le nombre de mois à ajouter à ou à soustraire de la `startdate`. Dans cet exemple, la valeur 2 est fournie.

Diagramme de la fonction `addmonths()`, exemple sans argument supplémentaire



La transaction 8193 a eu lieu le 14 août. Par conséquent, la fonction `addmonths()` renvoie le 14 octobre 2020 pour le champ `two_months_later`.

Exemple 2 – fin de mois relative

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions month-end en 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `relative_two_months_prior`, qui renvoie la date month-end relative tombant deux mois avant la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        addmonths(date,-2,1) as relative_two_months_prior
        ;
Load
*
Inline
[
id,date,amount
8188,'01/28/2022',37.23
8189,'01/31/2022',57.54
8190,'02/28/2022',17.17
8191,'04/29/2022',88.27
8192,'04/30/2022',57.42
8193,'05/31/2022',53.80
8194,'08/14/2022',82.06
8195,'10/07/2022',40.39
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

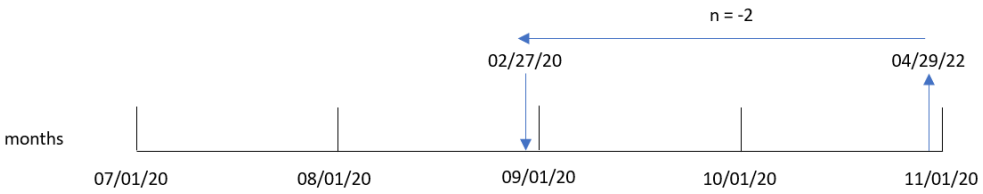
- `date`
- `relative_two_months_prior`

Tableau de résultats

date	relative_two_months_prior
01/28/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	02/27/2022
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

Le champ `relative_two_months_prior` est créé dans l'instruction LOAD précédente à l'aide de la fonction `addmonths()`. Le premier argument fourni identifie la date en cours d'évaluation. Le deuxième argument est le nombre de mois à ajouter à ou à soustraire de la `startdate`. Dans cet exemple, la valeur -2 est fournie. L'argument final est le mode, d'une valeur 1, qui force la fonction à calculer la date month-end relative pour toutes les dates ultérieures ou égales au 28.

Diagramme de la fonction `addmonths()`, exemple avec `n=-2`



La transaction 8191 a lieu le 29 avril 2022. Initialement, deux mois avant, cela correspondrait au mois de février. Ensuite, du fait que le troisième argument de la fonction définit le mode sur 1 et que la valeur day est ultérieure au 27, la fonction calcule la valeur month-end relative. La fonction identifie que le 29 est l'avant-dernier jour d'avril et renvoie par conséquent l'avant-dernier jour de février, le 27.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie la date tombant deux mois après la transaction est créé comme mesure dans un objet graphique.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=addmonths(date,2)
```

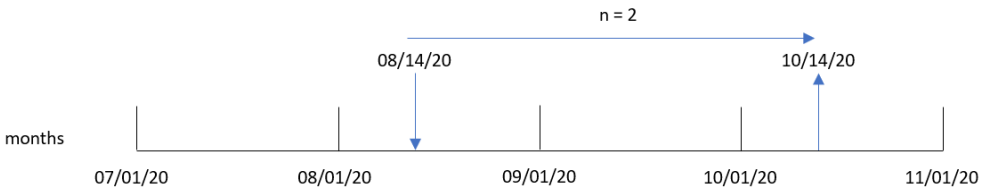
Tableau des résultats

date	=addmonths(date,2)
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020

date	=addmonths(date,2)
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

La mesure `two_months_later` est créée dans l'objet graphique à l'aide de la fonction `addmonths()`. Le premier argument fourni identifie la date en cours d'évaluation. Le deuxième argument est le nombre de mois à ajouter à ou à soustraire de la `startdate`. Dans cet exemple, la valeur 2 est fournie.

Diagramme de la fonction `addmonths()`, exemple objet graphique



La transaction 8193 a eu lieu le 14 août. Par conséquent, la fonction `addmonths()` renvoie le 14 octobre 2020 pour le champ `two_months_later`.

Exemple 4 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Mobile_Plans`.
- Informations avec l'ID de contrat, la date de début, la durée du contrat et les frais mensuels.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de contrat, la date de fin de chaque contrat téléphonique.

Script de chargement

```
Mobile_Plans:
Load
*
Inline
[
contract_id,start_date,contract_length,monthly_fee
8188,'01/13/2020',18,37.23
8189,'02/26/2020',24,17.17
8190,'03/27/2020',36,88.27
8191,'04/16/2020',24,57.42
8192,'05/21/2020',24,53.80
8193,'08/14/2020',12,82.06
8194,'10/07/2020',18,40.39
8195,'12/05/2020',12,87.21
8196,'01/22/2021',12,95.93
8197,'02/03/2021',18,45.89
8198,'03/17/2021',24,36.23
8199,'04/23/2021',24,25.66
8200,'05/04/2021',12,82.77
8201,'06/30/2021',12,69.98
8202,'07/26/2021',12,76.11
8203,'12/27/2021',36,25.12
8204,'06/06/2022',24,46.23
8205,'07/18/2022',12,84.21
8206,'11/14/2022',12,96.24
8207,'12/12/2022',18,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- contract_id
- start_date
- contract_length

Pour calculer la date de fin de chaque contrat, créez la mesure suivante :

```
=addmonths(start_date,contract_length, 0)
```

Tableau de résultats

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	01/22/2022
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024
8205	07/18/2022	12	07/18/2023
8206	11/14/2022	12	11/14/2023
8207	12/12/2022	18	06/12/2024

addyears

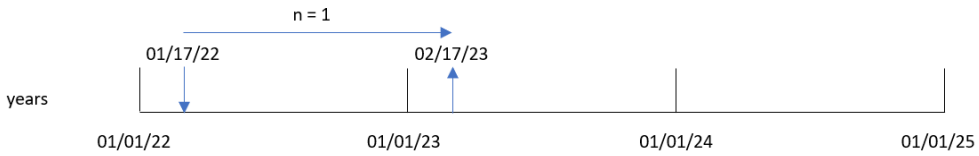
Cette fonction renvoie la date correspondant à **n** années après la date **startdate** ou, si **n** est une valeur négative, la date correspondant à **n** années avant la date **startdate**.

Syntaxe :

```
AddYears(startdate, n)
```

Type de données renvoyé : double

Exemple de diagramme de la fonction `addyears()`



La fonction `addyears()` ajoute ou soustrait un nombre défini d'années, `n`, d'une `startdate`. Elle renvoie ensuite la date obtenue.

Arguments

Argument	Description
startdate	Date de début sous forme d'horodatage, par exemple '2012-10-12'.
n	Nombre d'années sous forme d'entier positif ou négatif.

Exemples de fonction

Exemple	Résultat
`addyears ('01/29/2010',3)`	Renvoie '01/29/2013'.
`addyears ('01/29/2010',-1)`	Renvoie '01/29/2009'.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple simple

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `two_years_later`, qui renvoie la date tombant deux ans après la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        addyears(date,2) as two_years_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

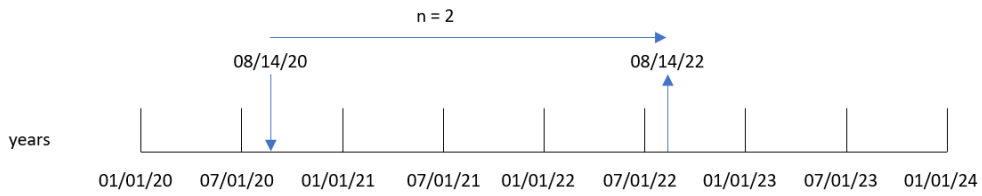
- `date`
- `two_years_later`

Tableau de résultats

date	two_years_later
01/10/2020	01/10/2022
02/28/2020	02/28/2022
04/09/2020	04/09/2022
04/16/2020	04/16/2022
05/21/2020	05/21/2022
08/14/2020	08/14/2022
10/07/2020	10/07/2022
12/05/2020	12/05/2022
01/22/2021	01/22/2023
02/03/2021	02/03/2023
03/17/2021	03/17/2023
04/23/2021	04/23/2023
05/04/2021	05/04/2023
06/30/2021	06/30/2023
07/26/2021	07/26/2023
12/27/2021	12/27/2023
02/02/2022	02/02/2024
02/26/2022	02/26/2024
03/07/2022	03/07/2024
03/11/2022	03/11/2024

Le champ `two_years_later` est créé dans l'instruction LOAD précédente à l'aide de la fonction `addyears()`. Le premier argument fourni identifie la date en cours d'évaluation. Le deuxième argument est le nombre d'années à ajouter à ou à soustraire de la date de début. Dans cet exemple, la valeur 2 est fournie.

Diagramme de la fonction addyears(), exemple de base



La transaction 8193 a eu lieu le 14 août 2020. Par conséquent, la fonction `addyears()` renvoie le 14 août 2022 pour le champ `two_years_later`.

Exemple 2 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

Dans un objet graphique, créez une mesure, `prior_year_date`, qui renvoie la date tombant un an avant la transaction.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
```

```
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer la date tombant un an avant chaque transaction, créez la mesure suivante :

```
=addyears(date,-1)
```

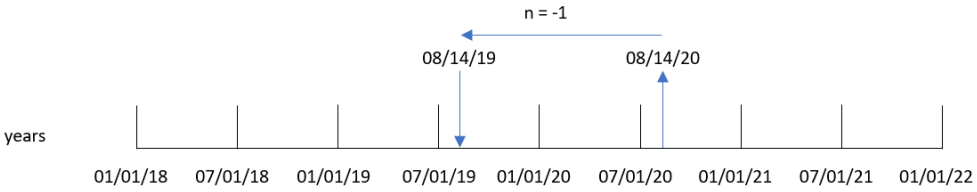
Tableau de résultats

date	=addyears(date,-1)
01/10/2020	01/10/2019
02/28/2020	02/28/2019
04/09/2020	04/09/2019
04/16/2020	04/16/2019
05/21/2020	05/21/2019
08/14/2020	08/14/2019
10/07/2020	10/07/2019
12/05/2020	12/05/2019
01/22/2021	01/22/2020
02/03/2021	02/03/2020
03/17/2021	03/17/2020
04/23/2021	04/23/2020
05/04/2021	05/04/2020
06/30/2021	06/30/2020
07/26/2021	07/26/2020
12/27/2021	12/27/2020
02/02/2022	02/02/2021
02/26/2022	02/26/2021

date	=addyears(date,-1)
03/07/2022	03/07/2021
03/11/2022	03/11/2021

La mesure `one_year_prior` est créée dans l'objet graphique à l'aide de la fonction `addyears()`. Le premier argument fourni identifie la date en cours d'évaluation. Le deuxième argument est le nombre d'années à ajouter à ou à soustraire de la `startdate`. Dans cet exemple, la valeur -1 est fournie.

Diagramme de la fonction `addyears()`, exemple objet graphique



La transaction 8193 a eu lieu le 14 août. Par conséquent, la fonction `addyears()` renvoie le 14 août 2019 pour le champ `one_year_prior`.

Exemple 3 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Warranties`.
- Information avec l'ID de produit, la date d'achat, la durée de la garantie et le prix d'achat.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de produit, la date de fin de garantie de chaque produit.

Script de chargement

```
Warranties:
Load
*
Inline
[
product_id,purchase_date,warranty_length,purchase_price
8188,'01/13/2020',4,32000
8189,'02/26/2020',2,28000
8190,'03/27/2020',3,41000
8191,'04/16/2020',4,17000
```

```
8192,'05/21/2020',2,25000
8193,'08/14/2020',1,59000
8194,'10/07/2020',2,12000
8195,'12/05/2020',3,12000
8196,'01/22/2021',4,24000
8197,'02/03/2021',1,50000
8198,'03/17/2021',2,80000
8199,'04/23/2021',3,10000
8200,'05/04/2021',4,30000
8201,'06/30/2021',3,30000
8202,'07/26/2021',4,20000
8203,'12/27/2021',4,10000
8204,'06/06/2022',2,25000
8205,'07/18/2022',1,32000
8206,'11/14/2022',1,30000
8207,'12/12/2022',4,22000
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `product_id`
- `purchase_date`
- `warranty_length`

Pour calculer la date de fin de garantie de chaque produit, créez la mesure suivante :

```
=addyears(purchase_date,warranty_length)
```

Tableau de résultats

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8188	01/13/2020	4	01/13/2024
8189	02/26/2020	2	02/26/2022
8190	03/27/2020	3	03/27/2023
8191	04/16/2020	4	04/16/2024
8192	05/21/2020	2	05/21/2022
8193	08/14/2020	1	08/14/2021
8194	10/07/2020	2	10/07/2022
8195	12/05/2020	3	12/05/2023
8196	01/22/2021	4	01/22/2025
8197	02/03/2021	1	02/03/2022
8198	03/17/2021	2	03/17/2023

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8199	04/23/2021	3	04/23/2024
8200	05/04/2021	4	05/04/2025
8201	06/30/2021	3	06/30/2024
8202	07/26/2021	4	07/26/2025
8203	12/27/2021	4	12/27/2025
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

age

La fonction **age** renvoie l'âge atteint à l'heure définie dans l'argument **timestamp** (en années) d'une personne née le jour défini par la valeur **date_of_birth**.

Syntaxe :

```
age(timestamp, date_of_birth)
```

Peut correspondre à une expression.

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
timestamp	Horodatage, ou expression aboutissant à un horodatage, jusqu'auquel le nombre d'années passées doit être calculé.
date_of_birth	Date de naissance de la personne dont l'âge est en cours de calcul. Peut correspondre à une expression.

Exemples et résultats :
Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de script

Exemple	Résultat
`age('25/01/2014', '29/10/2012')`	Renvoie 1.
`age('29/10/2014', '29/10/2012')`	Renvoie 2.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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;
```

La table résultante affiche les valeurs renvoyées par la fonction age pour chaque enregistrement de la table.

Table des résultats

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

Convertit un horodatage UTC ou GMT en heure locale sous la forme d'une valeur double. Il peut s'agir de n'importe quelle ville, n'importe quelle région ou n'importe quel fuseau horaire dans le monde.

Syntaxe :

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
timestamp	Horodatage, ou expression aboutissant à un horodatage, à convertir.
place	Lieu ou fuseau horaire issu de la table de lieux ou fuseaux horaires valides ci-dessous. Une autre solution pour définir l'heure locale consiste à utiliser GMT ou UTC. Les valeurs et plages de décalage horaire suivantes sont valides : • GMT • GMT-12:00 - GMT-01:00 • GMT+01:00 - GMT+14:00 • UTC • UTC-12:00 - UTC-01:00 • UTC+01:00 - UTC+14:00 *Vous pouvez uniquement appliquer des décalages horaires standard. Il est impossible d'utiliser un décalage horaire arbitraire, tel que GMT-04:27.*
ignore_dst	Définissez la valeur sur True pour ignorer l'heure d'été DST (Daylight Saving Time).

L'heure résultante est ajustée en fonction de l'heure d'été, sauf si le paramètre **ignore_dst** est défini sur True.

Lieux et fuseaux horaires valides

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo

A-C	D-K	L-R	S-Z
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan

A-C	D-K	L-R	S-Z
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Exemples et résultats :

Exemples de script

Exemple	Résultat
`ConvertToLocalTime('2007-11-10 23:59:00','Paris')`	Renvoie « 2007-11-11 00:59:00 » et la représentation d'horodatage interne correspondante.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Renvoie l'heure de la côte est nord-américaine, par exemple New York.
`ConvertToLocalTime(UTC(), 'GMT-05:00', True)`	Renvoie l'heure de la côte est nord-américaine, par exemple New York, sans l'ajustement en fonction de l'heure d'été.

day

Cette fonction renvoie un entier représentant le jour au cours duquel la fraction de l'**expression** est interprétée comme une date selon l'interprétation standard des nombres.

La fonction renvoie le jour de la semaine d'une date donnée. Elle est couramment utilisée pour dériver un champ Jour dans le cadre d'une dimension Calendrier.

Syntaxe :

```
day(expression)
```

Type de données renvoyé : entier

Exemples de fonction

Exemple	Résultat
`day( 1971-10-12 )`	renvoie 12.
`day( 35648 )`	renvoie 6, car 35648 = 1997-08-06.

Exemple 1 – Ensemble de données DateFormat (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système `DateFormat` est définie au format DD/MM/YYYY.
- Un chargement précédent qui crée un champ supplémentaire, appelé `day_of_month`, via la fonction `day()`.
- Un champ supplémentaire, appelé `long_date`, qui utilise la fonction `date()` pour exprimer le nom de mois complet.

Script de chargement

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

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        day(date) as day_of_month
Inline
[
date
03/11/2022
03/12/2022
03/13/2022
03/14/2022
03/15/2022
03/16/2022
03/17/2022
03/18/2022
03/19/2022
03/20/2022
03/21/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `long_date`
- `day_of_month`

Tableau des résultats

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	11-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Le jour du mois est correctement évalué par la fonction `day()` du script.

Exemple 2 – Dates ANSI (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système DateFormat DD/MM/YYYY est utilisée. Cependant, les dates incluses dans l'ensemble de données sont au format de date standard ANSI.
- Un chargement précédent qui crée un champ supplémentaire, appelé `day_of_month`, via la fonction `date()`.
- Un champ supplémentaire, appelé `long_date`, qui utilise la fonction `date()` pour exprimer la date avec le nom de mois complet.

Script de chargement

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       day(date) as day_of_month
```

```
Inline
[
date
2022-03-11
2022-03-12
2022-03-13
2022-03-14
2022-03-15
2022-03-16
2022-03-17
2022-03-18
2022-03-19
2022-03-20
2022-03-21
 ];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `long_date`
- `day_of_month`

Tableau des résultats

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	11-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Le jour du mois est correctement évalué par la fonction `day()` du script.

Exemple 3 – Dates non formatées (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système `DateFormat` DD/MM/YYYY est utilisée.
- Un chargement précédent qui crée un champ supplémentaire, appelé `day_of_month`, via la fonction `day()`.
- La date non formatée d'origine, appelée `unformatted_date`.
- Un champ supplémentaire, appelé `long_date`, utilisant la fonction `date()`, est utilisé pour convertir la date numérique en champ de date formatée.

Script de chargement

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

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      day(date) as day_of_month

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- unformatted_date
- long_date
- day_of_month

Tableau des résultats

unformatted_date	long_date	day_of_month
44868	03-November- 2022	3
44898	03-December- 2022	3
44928	02-January- 2023	2
44958	01-February- 2023	1
44988	03-March- 2023	3
45008	23-March- 2023	23
45018	02-April- 2023	2
45038	22-April- 2023	22
45048	02-May- 2023	2
45068	22-May- 2023	22
45078	01-June- 2023	1

Le jour du mois est correctement évalué par la fonction `day()` du script.

Exemple 4 – Calcul du mois d'expiration (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de commandes passées en mars appelé `Orders`. La table contient trois champs :
  - id
  - order_date
  - amount

Script de chargement

```
Orders:
Load
        id,
```

```
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `order_date`.

Pour calculer la date de livraison, créez la mesure suivante : `=day(order_date+5)`.

Tableau des résultats

order_date	=day(order_date+5)
03/11/2022	16
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
03/18/2022	23
03/19/2022	24
03/20/2022	25
03/21/2022	26

La fonction `day()` détermine correctement qu'une commande passée le 11 mars sera livrée le 16 en fonction d'une période de livraison de 5 jours.

dayend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du jour figurant dans **time**. Le format de sortie par défaut correspond à l'argument **TimestampFormat** défini dans le script.

Syntaxe :

```
DayEnd(time[, [period_no[, day_start]])
```

Cas d'utilisation

La fonction `dayend()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction du jour qui n'a pas encore eu lieu. Par exemple, pour calculer les dépenses totales non encore encourues au cours de la journée.

Type de données renvoyé : double

Arguments

Argument	Description
time	Horodatage à évaluer.
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique le jour contenant **time**. Les valeurs négatives de l'argument **period_no** indiquent les jours passés tandis que les valeurs positives désignent les jours à venir.
day_start	Pour spécifier que les jours ne commencent pas à minuit, indiquez un décalage sous forme de fraction d'un jour dans **day_start**. Par exemple, 0.125 signifie 3 h du matin. En d'autres termes, pour créer le décalage, divisez l'heure de début par 24 heures. Par exemple, pour qu'une journée commence à 7h00 du matin, utilisez la fraction 7/24.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
dayend('01/25/2013 16:45:00')	Returns 01/25/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', -1)	Renvoie 01/24/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', 0, 0.5)	Returns 01/26/2013 11:59:59. PM

Exemple 1 - Script de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant une liste de dates, qui est chargé dans une table nommée "Calendar".
- La variable système `DateFormat` par défaut `(MM/DD/YYYY)`.
- Un chargement précédent pour créer un champ supplémentaire, '`EOD_timestamp`', via la fonction `dayend()`.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
      date,
      dayend(date) as EOD_timestamp
      ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `EOD_timestamp`

Tableau de résultats

date	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 11:59:59 PM
03/12/2022 4:34:58 AM	3/12/2022 11:59:59 PM
03/13/2022 5:15:55 AM	3/13/2022 11:59:59 PM
03/14/2022 9:25:14 AM	3/14/2022 11:59:59 PM
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM
03/16/2022 10:44:42 AM	3/16/2022 11:59:59 PM
03/17/2022 11:33:30 AM	3/17/2022 11:59:59 PM
03/18/2022 12:58:14 PM	3/18/2022 11:59:59 PM
03/19/2022 4:23:12 PM	3/19/2022 11:59:59 PM
03/20/2022 6:42:15 PM	3/20/2022 11:59:59 PM
03/21/2022 7:41:16 PM	3/21/2022 11:59:59 PM

Comme vous pouvez le voir dans le tableau ci-dessus, l'horodatage de fin de journée est généré pour chaque date de votre ensemble de données. L'horodatage se présente au format de la variable système `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Vous allez charger un ensemble de données contenant des réservations de services dans un tableau nommé Services.

L'ensemble de données inclut les champs suivants :

- `service_id`
- `service_date`
- `amount`

Vous allez créer deux champs dans le tableau :

- `deposit_due_date` : Date à laquelle l'acompte doit être reçu. Il s'agit de la fin de la journée trois jours avant `service_date`.
- `final_payment_due_date` : Date à laquelle le paiement final doit être reçu. Il s'agit de la fin de la journée sept jours après `service_date`.

Les deux champs ci-dessus sont créés lors d'un chargement précédent via la fonction `dayend()` et fournissent les deux premiers paramètres, `time` et `period_no`.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3) as deposit_due_date,
     dayend(service_date,7) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Tableau de résultats

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 11:59:59 PM	3/18/2022 11:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 11:59:59 PM	3/19/2022 11:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 11:59:59 PM	3/20/2022 11:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 11:59:59 PM	3/21/2022 11:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 11:59:59 PM	3/22/2022 11:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 11:59:59 PM	3/23/2022 11:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 11:59:59 PM	3/24/2022 11:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 11:59:59 PM	3/25/2022 11:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 11:59:59 PM	3/26/2022 11:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 11:59:59 PM	3/27/2022 11:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 11:59:59 PM	3/28/2022 11:59:59 PM

Les valeurs des nouveaux champs se présentent au format `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Étant donné que la fonction `dayend()` a été utilisée, les valeurs d'horodatage indiquent toutes la dernière milliseconde de la journée.

Les valeurs de date d'échéance de l'acompte sont trois jours avant la date du service, car le deuxième argument transmis dans la fonction `dayend()` est négatif.

Les valeurs de date d'échéance du paiement final sont sept jours après la date du service, car le deuxième argument transmis dans la fonction `dayend()` est positif.

Exemple 3 – script day_start

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous dans un nouvel onglet.

L'ensemble de données et le scénario utilisés dans cet exemple sont les mêmes que ceux de l'exemple précédent.

Comme dans l'exemple précédent, vous allez créer deux champs :

- `deposit_due_date` : Date à laquelle l'acompte doit être reçu. Il s'agit de la fin de la journée trois jours avant `service_date`.
- `final_payment_due_date` : Date à laquelle le paiement final doit être reçu. Il s'agit de la fin de la journée sept jours après `service_date`.

Cependant, votre entreprise souhaite appliquer une politique stipulant que la journée de travail commence à 17h00 et se termine à 17h00 le jour suivant. Votre entreprise peut alors surveiller les transactions qui ont lieu au cours de ces heures de travail.

Pour remplir ces conditions, les deux champs ci-dessus sont créés lors d'un chargement précédent via la fonction `dayend()` et fournissent l'ensemble des trois arguments, `time`, `period_no` et `day_start`.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3,17/24) as deposit_due_date,
     dayend(service_date,7,17/24) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Tableau de résultats

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 4:59:59 PM	3/18/2022 4:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 4:59:59 PM	3/19/2022 4:59:59 PM

service_date	deposit_due_date	final_payment_due_date
03/13/2022 10:44:42 AM	3/10/2022 4:59:59 PM	3/20/2022 4:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 4:59:59 PM	3/21/2022 4:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 4:59:59 PM	3/22/2022 4:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 4:59:59 PM	3/23/2022 4:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 4:59:59 PM	3/24/2022 4:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 4:59:59 PM	3/25/2022 4:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 4:59:59 PM	3/26/2022 4:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 4:59:59 PM	3/27/2022 4:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 4:59:59 PM	3/28/2022 4:59:59 PM

Même si les dates restent les mêmes que celles de l'exemple 2, leur horodatage est maintenant la dernière milliseconde avant 17h00, car la valeur du troisième argument, `day_start`, transmise à la fonction `dayend ()`, est 17/24.

Exemple 4 – Exemple de graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

L'ensemble de données et le scénario utilisés dans cet exemple sont les mêmes que ceux des deux exemples précédents. L'entreprise souhaite appliquer une politique stipulant que la journée de travail commence à 17h00 et se termine à 17h00 le jour suivant.

Comme dans l'exemple précédent, vous allez créer deux champs :

- `deposit_due_date` : Date à laquelle l'acompte doit être reçu. Il s'agit de la fin de la journée trois jours avant `service_date`.
- `final_payment_due_date` : Date à laquelle le paiement final doit être reçu. Il s'agit de la fin de la journée sept jours après `service_date`.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
Load
service_id,
service_date,
amount
Inline
[
```

```
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

`service_date`.

Pour créer le champ `deposit_due_date`, créez la mesure suivante :

`=dayend(service_date,-3,17/24)`.

Ensuite, pour créer le champ `final_payment_due_date`, créez la mesure suivante :

`=dayend(service_date,7,17/24)`.

Tableau de résultats

service_date	=dayend(service_date,-3,17/24)	=dayend(service_date,7,17/24
03/11/2022	3/8/2022 16:59:59 PM	3/18/2022 16:59:59 PM
03/12/2022	3/9/2022 16:59:59 PM	3/19/2022 16:59:59 PM
03/13/2022	3/10/2022 16:59:59 PM	3/20/2022 16:59:59 PM
03/14/2022	3/11/2022 16:59:59 PM	3/21/2022 16:59:59 PM
03/15/2022	3/12/2022 16:59:59 PM	3/22/2022 16:59:59 PM
03/16/2022	3/13/2022 16:59:59 PM	3/23/2022 16:59:59 PM
03/17/2022	3/14/2022 16:59:59 PM	3/24/2022 16:59:59 PM
03/18/2022	3/15/2022 16:59:59 PM	3/25/2022 16:59:59 PM
03/19/2022	3/16/2022 16:59:59 PM	3/26/2022 16:59:59 PM
03/20/2022	3/17/2022 16:59:59 PM	3/27/2022 16:59:59 PM
03/21/2022	3/18/2022 16:59:59 PM	3/28/2022 16:59:59 PM

Les valeurs des nouveaux champs se présentent au format `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Étant donné que la fonction `dayend()` a été utilisée, les valeurs d'horodatage indiquent toutes la dernière milliseconde de la journée.

Les valeurs de date d'échéance du paiement sont trois jours avant la date du service, car le deuxième argument transmis dans la fonction `dayend()` est négatif.

Les valeurs de date d'échéance du paiement final sont sept jours après la date du service, car le deuxième argument transmis dans la fonction `dayend()` est positif.

L'horodatage des dates est la dernière milliseconde avant 17h00, car la valeur du troisième argument, `day_start`, transmise dans la fonction `dayend()`, est 17/24.

daylightsaving

Renvoie le réglage actif de l'heure d'été, tel qu'il est défini dans Windows.

Syntaxe :

```
DaylightSaving( )
```

Type de données renvoyé : double

Exemple :

```
daylightsaving( )
```

dayname

Cette fonction renvoie une valeur affichant la date avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du jour contenant l'argument **time**.

Syntaxe :

```
DayName(time[, period_no [, day_start]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
time	Horodatage à évaluer.
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique le jour contenant **time**. Les valeurs négatives de l'argument **period_no** indiquent les jours passés tandis que les valeurs positives désignent les jours à venir.
day_start	Pour spécifier que les jours ne commencent pas à minuit, indiquez un décalage sous forme de fraction d'un jour dans **day_start**. Par exemple, 0.125 signifie 3 h du matin.

Exemples et résultats :
Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de script

Exemple	Résultat
`dayname('25/01/2013 16:45:00')`	Renvoie 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Renvoie 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Renvoie 25/01/2013. L'affichage de l'horodatage complet montre que la valeur numérique sous-jacente correspond à '25/01/2013 12:00:00.000.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

Dans cet exemple, le nom du jour est créé à partir de l'horodatage marquant le début de la journée après chaque date de facture dans la table.

```
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;
```

La table résultante présente les dates initiales et une colonne contenant la valeur de retour de la fonction dayname(). Vous pouvez afficher l'horodatage complet en spécifiant le formatage dans le panneau des propriétés.

Table des résultats

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

Cette fonction calcule le numéro de jour du trimestre dans lequel tombe un horodatage. Cette fonction est utilisée lors de la création d'un Calendrier principal.

Syntaxe :

```
DayNumberOfQuarter(timestamp[,start_month])
```

Type de données renvoyé : entier

Arguments

Argument	Description
timestamp	Date ou horodatage à évaluer.
start_month	Si vous spécifiez un argument **start_month** compris entre 2 et 12 (1 si l'argument est omis), il se peut que le début de l'année soit avancé au premier jour de n'importe quel mois. Par exemple, si vous voulez travailler sur un exercice fiscal débutant le premier mars, spécifiez **start_month** = 3.

Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`DayNumberOfQuarter('12/09/2014')`	Renvoie 74, le numéro de jour du trimestre actuel.
`DayNumberOfQuarter ('12/09/2014',3)`	Renvoie 12, le numéro de jour du trimestre actuel. Dans ce cas, le premier trimestre commence au mois de mars (car start_month est défini sur 3). Cela signifie donc que le trimestre actuel est le troisième, qui a débuté le 1er septembre.

Exemple 1 – Janvier début de l'année (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un simple ensemble de données contenant une liste de dates, qui est chargé dans une table nommée `Calendar`. La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- Un chargement précédent qui crée un champ supplémentaire, appelé `DayNrQtr`, via la fonction `DayNumberOfQuarter()`.

Outre la date, aucun paramètre supplémentaire n'est fourni à la fonction.

Script de chargement

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

Calendar:
Load
        date,
        DayNumberOfQuarter(date) as DayNrQtr
        ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
```

```
04/01/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `daynrqtr`

Tableau des résultats

date	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Le premier jour de l'année est le 1er janvier, car aucun deuxième argument n'a été transmis dans la fonction `DayNumberOfQuarter()`.

Le 1er janvier est le premier jour du trimestre, tandis que le 1er février est le 32e jour du trimestre. Le 31 mars est le 91e et dernier jour du trimestre, tandis que le 1er avril est le premier jour du deuxième trimestre.

Exemple 2 – Février début de l'année (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.

- Un argument `start_month` commençant le 1er février. Cela définit le début de l'exercice financier au 1er février.

Script de chargement

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

Calendar:
Load
       date,
       DayNumberOfQuarter(date,2) as DayNrQtr
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `daynrqtr`

Tableau des résultats

date	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Le premier jour de l'année est le 1er février, car le deuxième argument transmis dans la fonction `DayNumberOfQuarter()` était 2.

Le premier trimestre de l'année se déroule de février à avril, tandis que le quatrième trimestre se déroule de novembre à janvier. Cela apparaît dans le tableau de résultats, dans lequel le 1er février est le premier jour du trimestre, tandis que le 31 janvier est le 92e et dernier jour du trimestre.

Exemple 3 – Janvier début de l'année (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. La valeur du jour du trimestre est calculée via une mesure dans un objet graphique.

Script de chargement

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

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=daynumberofquarter(date)
```

Tableau des résultats

date	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Le premier jour de l'année est le 1er janvier, car aucun deuxième argument n'a été transmis dans la fonction `DayNumberOfQuarter()`.

Le 1er janvier est le premier jour du trimestre, tandis que le 1er février est le 32e jour du trimestre. Le 31 mars est le 91e et dernier jour du trimestre, tandis que le 1er avril est le premier jour du deuxième trimestre.

Exemple 4 – Février début de l'année (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- L'exercice financier se déroule du 1er février au 31 janvier.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. La valeur du jour du trimestre est calculée via une mesure dans un objet graphique.

Script de chargement

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

Calendar:
Load
date
Inline
```

```
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Objet graphique

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=daynumberofquarter(date,2)
```

Résultats

Tableau des résultats

date	=daynumberofquarter(date,2)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Le premier jour de l'année est le 1er janvier, car le deuxième argument transmis dans la fonction `DayNumberOfQuarter()` était 2.

Le premier trimestre de l'année se déroule de février à avril, tandis que le quatrième trimestre se déroule de novembre à janvier. Cela apparaît dans le tableau de résultats, dans lequel le 1er février est le premier jour du trimestre, tandis que le 31 janvier est le 92e et dernier jour du trimestre.

daynumberofyear

Cette fonction calcule le numéro de jour de l'année dans lequel tombe un horodatage. Le calcul est effectué à partir de la première milliseconde du premier jour de l'année, mais le premier mois peut être décalé.

Syntaxe :

```
DayNumberOfYear(timestamp[,start_month])
```

Type de données renvoyé : entier

Arguments

Argument	Description
timestamp	Date ou horodatage à évaluer.
start_month	Si vous spécifiez un argument **start_month** compris entre 2 et 12 (1 si l'argument est omis), il se peut que le début de l'année soit avancé au premier jour de n'importe quel mois. Par exemple, si vous voulez travailler sur un exercice fiscal débutant le premier mars, spécifiez **start_month** = 3.

Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`DayNumberOfYear( '12/09/2014' )`	Renvoie 256, le numéro de jour calculé à partir du premier jour de l'année.
`DayNumberOfYear( '12/09/2014',3 )`	Renvoie 196, le numéro de jour calculé à partir du 1er mars.

Exemple 1 – Janvier début de l'année (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un simple ensemble de données contenant une liste de dates, qui est chargé dans une table nommée `Calendar`. La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.

- Un chargement précédent qui crée un champ supplémentaire, appelé `daynryear`, via la fonction `DayNumberOfYear()`.

Outre la date, aucun paramètre supplémentaire n'est fourni à la fonction.

Script de chargement

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

Calendar:
Load
       date,
       DayNumberOfYear(date) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `daynryear`

Tableau de résultats

date	daynryear
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182

date	daynryear
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Le premier jour de l'année est le 1er janvier, car aucun deuxième argument n'a été transmis dans la fonction `DayNumberOfYear()`.

Le 1er janvier est le premier jour du trimestre, tandis que le 1er février est le 32e jour de l'année. Le 30 juin est le 182e, tandis que le 31 décembre est le 366e et dernier jour de l'année.

Exemple 2 – Novembre début de l'année (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- Un argument `start_month` commençant le 1er novembre. Cela définit le début de l'exercice financier au 1er novembre.

Script de chargement

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

Calendar:
Load
       date,
       DayNumberOfYear(date,11) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
```

```
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `daynryear`

Tableau des résultats

date	daynryear
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Le premier jour de l'année est le 1er novembre, car le deuxième argument transmis dans la fonction `DayNumberOfYear()` était 11.

Le 1er janvier est le premier jour du trimestre, tandis que le 1er février est le 32e jour de l'année. Le 30 juin est le 182e, tandis que le 31 décembre est le 366e et dernier jour de l'année.

Exemple 3 – Janvier début de l'année (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. La valeur du jour du trimestre est calculée via une mesure dans un objet graphique.

Script de chargement

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

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=daynumberofyear(date)
```

Tableau des résultats

date	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305

date	=daynumberofyear(date)
11/01/2022	306
12/31/2022	366

Le premier jour de l'année est le 1er janvier, car aucun deuxième argument n'a été transmis dans la fonction `DayNumberOfYear()`.

Le 1er janvier est le premier jour de l'année, tandis que le 1er février est le 32e jour de l'année. Le 30 juin est le 182e, tandis que le 31 décembre est le 366e et dernier jour de l'année.

Exemple 4 – Novembre début de l'année (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- L'exercice financier se déroule du 1er novembre au 31 octobre.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. La valeur du jour de l'année est calculée via une mesure dans un objet graphique.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=daynumberofyear(date)
```

Tableau des résultats

date	=daynumberofyear(date,11)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Le premier jour de l'année est le 1er novembre, car le deuxième argument transmis dans la fonction `DayNumberOfYear()` était 11.

L'exercice financier se déroule entre novembre et octobre. Cela apparaît dans le tableau de résultats, dans lequel le 1er novembre est le premier jour de l'année, tandis que le 31 octobre est le 366e et dernier jour de l'année.

daystart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du jour figurant dans l'argument **time**. Le format de sortie par défaut correspond à l'argument **TimestampFormat** défini dans le script.

Syntaxe :

```
DayStart(time[, [period_no[, day_start]])
```

Type de données renvoyé : double

Arguments

Argument	Description
time	Horodatage à évaluer.

Argument	Description
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique le jour contenant **time**. Les valeurs négatives de l'argument **period_no** indiquent les jours passés tandis que les valeurs positives désignent les jours à venir.
day_start	Pour spécifier que les jours ne commencent pas à minuit, indiquez un décalage sous forme de fraction d'un jour dans **day_start**. Par exemple, 0.125 signifie 3 h du matin. En d'autres termes, pour créer le décalage, divisez l'heure de début par 24 heures. Par exemple, pour qu'une journée commence à 7h00 du matin, utilisez la fraction 7/24.

Cas d'utilisation

La fonction `daystart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction du jour qui s'est écoulée jusqu'à présent. Par exemple, cela peut permettre de calculer le total des salaires touchés par les employés jusqu'à ce jour.

Ces exemples utilisent le format d'horodatage `'M/D/YYYY h:mm:ss[.fff] TT'`. Le format d'horodatage est indiqué dans l'instruction `SET TimeStamp` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`daystart('01/25/2013 4:45:00 PM')`	Renvoie 1/25/2013 12:00:00 AM.
`daystart('1/25/2013 4:45:00 PM', -1)`	Renvoie 1/24/2013 12:00:00 AM.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Renvoie 1/25/2013 12:00:00 PM.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - exemple simple

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données simple contenant une liste de dates, qui est chargé dans une table nommée `Calendar`.
- La variable système `TimeStampFormat` par défaut (`(M/D/YYYY h:mm:ss[.fff] TT`) est utilisée.
- Chargement précédent qui crée un champ supplémentaire, appelé `SOD_timestamp`, via la fonction `daystart()`.

Outre la date, aucun paramètre supplémentaire n'est fourni à la fonction.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
        date,
        daystart(date) as SOD_timestamp
        ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- SOD_timestamp

Tableau de résultats

date	SOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 12:00:00 AM
03/12/2022 4:34:58 AM	3/12/2022 12:00:00 AM
03/13/2022 5:15:55 AM	3/13/2022 12:00:00 AM
03/14/2022 9:25:14 AM	3/14/2022 12:00:00 AM
03/15/2022 10:06:54 AM	3/15/2022 12:00:00 AM
03/16/2022 10:44:42 AM	3/16/2022 12:00:00 AM
03/17/2022 11:33:30 AM	3/17/2022 12:00:00 AM
03/18/2022 12:58:14 PM	3/18/2022 12:00:00 AM
03/19/2022 4:23:12 PM	3/19/2022 12:00:00 AM
03/20/2022 6:42:15 PM	3/20/2022 12:00:00 AM
03/21/2022 7:41:16 PM	3/21/2022 12:00:00 AM

Comme le montre le tableau ci-dessus, l'horodatage de fin de journée est généré pour chaque date de notre ensemble de données. L'horodatage se présente au format de la variable système `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des amendes de stationnement, qui est chargé dans une table nommée `Fines`. L'ensemble de données inclut les champs suivants :
  - id
  - due_date
  - number_plate
  - amount

- Chargement précédent utilisant la fonction `daystart()` et fournissant l'ensemble des trois paramètres suivants : `time`, `period_no` et `day_start`. Ce chargement précédent crée les deux nouveaux champs de date suivants :
    - Un champ de date `early_repayment_period`, qui commence sept jours avant la date d'échéance de paiement.
    - Un champ de date `late_penalty_period`, qui commence quatorze jours après la date d'échéance de paiement.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
        *,
        daystart(due_date,-7) as early_repayment_period,
        daystart(due_date,14) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `due_date`
- `early_repayment_period`
- `late_penalty_period`

Tableau de résultats

due_date	early_repayment_period	late_penalty_period
02/11/2022 9:25:14 AM	2/4/2022 12:00:00 AM	2/25/2022 12:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 12:00:00 AM	4/8/2022 12:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 12:00:00 AM	4/28/2022 12:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 12:00:00 AM	7/12/2022 12:00:00 AM

due_date	early_repayment_period	late_penalty_period
08/15/2022 12:58:14 PM	8/8/2022 12:00:00 AM	8/29/2022 12:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 12:00:00 AM	11/30/2022 12:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 12:00:00 AM	1/31/2023 12:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 12:00:00 AM	4/5/2023 12:00:00 AM

Les valeurs des nouveaux champs se présentent au format `TimestampFormat M/DD/YYYY tt`. Étant donné que la fonction `daystart()` a été utilisée, les valeurs d'horodatage indiquent toutes la première milliseconde de la journée.

Les valeurs de période de remboursement anticipé sont sept jours avant la date d'échéance, car le deuxième argument transmis dans la fonction `daystart()` est négatif.

Les valeurs de période de remboursement tardif sont quatorze jours après la date d'échéance, car le deuxième argument transmis dans la fonction `daystart()` est positif.

Exemple 3 - day_start

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux de l'exemple précédent.
- Même chargement précédent que celui de l'exemple précédent.

Dans cet exemple, nous décidons que la journée de travail commence et se termine tous les jours à 7h00 le matin.

Script de chargement

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

Fines:
    Load
        *,
        daystart(due_date,-7,7/24) as early_repayment_period,
        daystart(due_date,14, 7/24) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
```

```
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- due_date
- early_repayment_period
- late_penalty_period

Tableau de résultats

due_date	early_repayment_period	late_penalty_period
02/11/2022	2/3/2022 7:00:00 AM	2/24/2022 7:00:00 AM
03/25/2022	3/17/2022 7:00:00 AM	4/7/2022 7:00:00 AM
04/14/2022	4/6/2022 7:00:00 AM	4/27/2022 7:00:00 AM
06/28/2022	6/20/2022 7:00:00 AM	7/11/2022 7:00:00 AM
08/15/2022	8/7/2022 7:00:00 AM	8/28/2022 7:00:00 AM
11/16/2022	11/8/2022 7:00:00 AM	11/29/2022 7:00:00 AM
01/17/2023	1/9/2023 7:00:00 AM	1/30/2023 7:00:00 AM
03/22/2023	3/14/2023 7:00:00 AM	4/4/2023 7:00:00 AM

L'horodatage des dates est maintenant de 7h00 du matin, car la valeur de l'argument `day_start` transmise dans la fonction `daystart()` était 7/24. Cela fixe le début de la journée à 7h00 du matin.

Étant donné que le champ `due_date` n'a pas d'horodatage, il est traité comme minuit (12:00 AM), qui fait par conséquent encore partie de la journée précédente, puisque les journées commencent et se terminent à 7h00 du matin. Par conséquent, la période de remboursement anticipé d'une amende due le 11 février commence le 3 février à 7h00 du matin.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Cet exemple utilise le même ensemble de données et le même scénario que ceux de l'exemple précédent.

Cependant, seule la table `Fines` d'origine est chargée dans l'application, avec les deux valeurs de dates d'échéance supplémentaires calculées dans un objet graphique.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
*
Inline
[
id, due_date, numer_plate,amount
1,02/11/2022 9:25:14 AM, 573RJG,50.00
2,03/25/2022 10:06:54 AM, SC41854,50.00
3,04/14/2022 10:44:42 AM, 8EHZ378,50.00
4,06/28/2022 11:33:30 AM, 8HSS198,50.00
5,08/15/2022 12:58:14 PM, 1221665,50.00
6,11/16/2022 4:23:12 PM, EAK473,50.00
7,01/17/2023 6:42:15 PM, KD6822,50.00
8,03/22/2023 7:41:16 PM, 1GGLB,50.00
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `due_date`.
2. Pour créer le champ `early_repayment_period`, créez la mesure suivante : `=daystart(due_date,-7,7/24)`
3. Pour créer le champ `late_penalty_period`, créez la mesure suivante : =daystart(due_date,14,7/24)

Tableau de résultats

due_date	=daystart(due_date,-7,7/24)	=daystart(due_date,14,7/24)
02/11/2022 9:25:14 AM	2/4/2022 7:00:00 AM	2/25/2022 7:00:00 AM

due_date	=daystart(due_date,-7,7/24)	=daystart(due_date,14,7/24)
03/25/2022 10:06:54 AM	3/18/2022 7:00:00 AM	4/8/2022 7:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 7:00:00 AM	4/28/2022 7:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 7:00:00 AM	7/12/2022 7:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 7:00:00 AM	8/29/2022 7:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 7:00:00 AM	11/30/2022 7:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 7:00:00 AM	1/31/2023 7:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 7:00:00 AM	4/5/2023 7:00:00 AM

Les valeurs des nouveaux champs se présentent au format `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Étant donné que la fonction `daystart()` a été utilisée, les valeurs d'horodatage correspondent à la première milliseconde de la journée.

Les valeurs de période de remboursement anticipé sont sept jours avant la date d'échéance, car le deuxième argument transmis dans la fonction `daystart()` était négatif.

Les valeurs de période de remboursement tardif sont quatorze jours après la date d'échéance, car le deuxième argument transmis dans la fonction `daystart()` était positif.

L'horodatage des dates est de 7h00 du matin, car la valeur du troisième argument transmise dans la fonction `daystart()`, `day_start`, était 7/24.

firstworkdate

La fonction **firstworkdate** renvoie la date de début la plus récente pour atteindre la valeur **no_of_workdays** (du lundi au vendredi) se terminant au plus tard à la date définie par la valeur **end_date**, en tenant compte des jours de congé facultatifs indiqués. Les valeurs des arguments **end_date** et **holiday** doivent correspondre à des dates ou à des horodatages valides.

Syntaxe :

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
end_date	Horodatage de la date de fin à évaluer.
no_of_workdays	Nombre de jours ouvrables à atteindre.

Argument	Description
holiday	Périodes de congé à exclure des jours ouvrables. Un congé est indiqué sous forme de date constante de type chaîne. Vous pouvez spécifier plusieurs dates de congé si vous les séparez par des virgules. **Exemple :** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exemples et résultats :
Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de script

Exemple	Résultat
`firstworkdate ('29/12/2014', 9)`	Renvoie '17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Renvoie 15/12/2014, car une période de congé de deux jours est prise en compte.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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;
```

La table résultante affiche les valeurs renvoyées par la fonction FirstWorkDate pour chaque enregistrement de la table.

Table des résultats

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014

InvID	EndDate	StartDate
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

Cette fonction renvoie l'heure Greenwich Mean Time actuelle, indiquée par les paramètres régionaux. La fonction renvoie des valeurs au format de variable système `TimestampFormat`.

À chaque chargement de l'application, tout objet graphique, table de script de chargement ou variable qui utilise la fonction `GMT` sera ajusté à la dernière heure GMT (Greenwich Mean Time - Heure moyenne de Greenwich) actuelle dérivée de l'horloge système.

Syntaxe :

```
GMT( )
```

Type de données renvoyé : double

Ces exemples utilisent le format d'horodatage `M/D/YYYY h:mm:ss[.fff] TT`. Le format de date est indiqué dans l'instruction `SET TimestampFormat` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`GMT()`	3/28/2022 2:47:36 PM

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - variable (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet. Cet exemple définira l'heure GMT (Greenwich Mean Time - Heure moyenne de Greenwich) actuelle comme variable dans le script de chargement via la fonction GMT.

Script de chargement

```
LET vGMT = GMT();
```

Résultats

Chargez les données et créez une feuille. À l'aide de l'objet graphique **Texte et image**, créez une zone de texte.

Ajoutez cette mesure à la zone de texte :

```
=vGMT
```

La zone de texte doit contenir une ligne de texte avec une date et une heure, comme illustré ci-dessous :

```
3/28/2022 2:47:36 PM
```

Exemple 2 - novembre début de l'année (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des livres de bibliothèque à rendre dont la date d'échéance est dépassée, qui est chargé dans une table nommée Overdue. La variable système DateFormat par défaut MM/DD/YYYY est utilisée.
- Création d'un nouveau champ appelé days_overdue, qui calcule le nombre de jours de retard de chaque livre.

Script de chargement

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

Overdue:
    Load
        *,
```

```
        Floor(GMT()-due_date) as days_overdue
        ;
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- due_date
- book_id
- days_overdue

Tableau de résultats

due_date	book_id	days_overdue
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Les valeurs du champ `days_overdue` sont calculées en recherchant la différence entre l'heure GMT (Greenwich Mean Time - Heure moyenne de Greenwich) actuelle, via la fonction `GMT()`, et la date d'échéance d'origine. Pour calculer uniquement les jours, les résultats sont arrondis au nombre entier le plus proche via la fonction `Floor()`.

Exemple 3 - objet graphique (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet. Le script de chargement contient le même ensemble de données que celui de l'exemple précédent. La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. La valeur du nombre de jours de retard est calculée via une mesure dans un objet graphique.

Script de chargement

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

Overdue:
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `due_date`
- `book_id`

Créez la mesure suivante :

```
=Floor(GMT() - due_date)
```

Tableau de résultats

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455

due_date	book_id	=Floor(GMT()-due_date)
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Les valeurs du champ `days_overdue` sont calculées en recherchant la différence entre l'heure GMT (Greenwich Mean Time - Heure moyenne de Greenwich) actuelle, via la fonction `GMT()`, et la date d'échéance d'origine. Pour calculer uniquement les jours, les résultats sont arrondis au nombre entier le plus proche via la fonction `Floor()`.

hour

Cette fonction renvoie un entier représentant l'heure au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

Syntaxe :

```
hour(expression)
```

Type de données renvoyé : entier

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`hour( '09:14:36' )`	La chaîne de texte fournie est implicitement convertie en horodatage, car elle correspond au format d'horodatage défini dans la variable TimestampFormat. L'expression renvoie 9.
`hour( '0.5555' )`	L'expression renvoie 13 (car 0.5555 = 13:19:55).

Exemple 1 – Variable (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant des transactions par horodatage.
- La variable système `TimeStamp` par défaut `(M/D/YYYY h:mm:ss[.fff] TT)`.

Créez un champ, '`hour`', calculant le moment où les achats ont eu lieu.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
  Load
    *,
    hour(date) as hour
    ;
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `hour`

Tableau de résultats

date	hour
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Les valeurs du champ hour sont créées via la fonction `hour()` et en transmettant la date comme expression dans l'instruction LOAD précédente.

Exemple 2 – Objet graphique (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- La variable système `TimeStamp` par défaut `(M/D/YYYY h:mm:ss[.fff] TT)`.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Les valeurs '`hour`' sont calculées via une mesure dans un objet graphique.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';
```

```
Transactions:
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer '`hour`', créez la mesure suivante :

```
=hour(date)
```

Tableau des résultats

due_date	=hour(date)
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Les valeurs de '`hour`' sont créées via la fonction `hour()` et en transmettant la date comme expression dans une mesure de l'objet graphique.

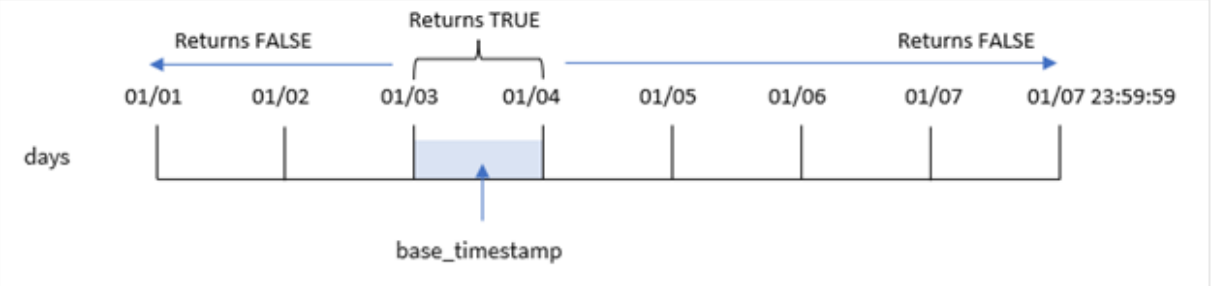
inday

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le jour comprenant l'argument **base_timestamp**.

Syntaxe :

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Diagramme de la fonction `inday`



La fonction `inday()` utilise l'argument `base_timestamp` pour identifier le jour de l'horodatage. L'heure de début du jour est, par défaut, minuit ; mais vous pouvez modifier l'heure de début du jour via l'argument `day_start` de la fonction `inday()`. Une fois ce jour défini, la fonction renverra des résultats booléens lors de la comparaison des valeurs d'horodatage prescrites au jour en question.

Cas d'utilisation

La fonction `inday()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. Cela renvoie une agrégation ou un calcul, suivant qu'une date évaluée s'est produite ou non au cours du jour de l'horodatage en question.

Par exemple, la fonction `inday()` peut être utilisée pour identifier l'ensemble des équipements fabriqués un jour donné.

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Arguments

Argument	Description
`timestamp`	Date et heure à comparer à `base_timestamp`.
`base_timestamp`	Date et heure utilisées pour évaluer l'horodatage.
`period_no`	Il est possible de décaler le jour à l'aide de l'argument `period_no`. `period_no` est un entier, où la valeur 0 indique le jour comprenant l'argument `base_timestamp`. Les valeurs négatives de l'argument `period_no` indiquent les jours passés tandis que les valeurs positives désignent les jours à venir.

Argument	Description
day_start	Si vous souhaitez utiliser des jours qui ne commencent pas à minuit, indiquez un décalage sous forme de fraction de jour dans l'argument day_start, par exemple 0.125 pour indiquer 3 heures du matin.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction SET DateFormat de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Renvoie True
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Renvoie False
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Renvoie False
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Renvoie True
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Renvoie False
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Renvoie True

Exemple 1 – Instruction LOAD (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant des transactions par horodatage chargé dans une table appelée Transactions.

- Un champ date fourni dans la variable système `TimeStamp` au format `(M/D/YYYY h:mm:ss[.fff] TT)`.
- Un chargement précédent contenant la fonction `inday()` définie sous forme de champ `in_day`.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_day`

Tableau de résultats

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1

date	in_day
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Le champ `in_day` est créé dans l'instruction LOAD précédente via la fonction `inday()` et en transmettant le champ date, un horodatage codé en dur pour le 5 janvier et une valeur `period_no` égale à `0` comme arguments de la fonction.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le script de chargement utilise le même ensemble de données et le même scénario que ceux utilisés dans le premier exemple.

Cependant, dans cet exemple, la tâche consiste à calculer si la date de transaction a eu lieu deux jours avant le 5 janvier.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', -2) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- in_day

Tableau de résultats

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-1
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	0
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Dans cet instance, étant donné que la valeur `period_no` de `-2` est utilisée comme argument de décalage dans la fonction `inday()`, la fonction détermine si chaque date de transaction a eu lieu le 3 janvier. Cela peut être vérifié dans le tableau de résultats, dans lequel une transaction renvoie un résultat booléen égal à TRUE.

Exemple 3 – day_start

Script de chargement et résultats

Vue d'ensemble

Le script de chargement utilise le même ensemble de données et le même scénario que ceux utilisés dans les exemples précédents.

Cependant, dans cet exemple, la politique de l'entreprise détermine que la journée de travail commence et se termine à 7h00 du matin.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0, 7/24) as in_day
        ;
Load
*
Inline
```

```
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_day`

Tableau de résultats

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	-1
01/04/2022 10:58:34 PM	-1
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Étant donné que l'argument `start_day` de 7/24, qui est 7h00 du matin, est utilisé dans la fonction `inday()`, la fonction détermine si chaque date de transaction a eu lieu le 4 janvier à partir de 7h00 du matin et le 5 janvier avant 7h00 du matin.

Cela peut être vérifié dans le tableau de résultats dans lequel les transactions qui ont lieu après 7h00 du matin le 4 janvier renvoient un résultat booléen égal à TRUE, tandis que les transactions qui ont lieu après 7h00 du matin le 5 janvier renvoient un résultat booléen égal à FALSE.

Exemple 4 – Objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le script de chargement utilise le même ensemble de données et le même scénario que ceux utilisés dans les exemples précédents.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Vous allez effectuer un calcul pour déterminer si une transaction a lieu le 5 janvier en créant une mesure dans un objet graphique.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- date

Pour calculer si une transaction a lieu le 5 janvier, créez la mesure suivante :

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Tableau de résultats

date	inday(date,'01/05/2022 12:00:00 AM',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0

date	inday(date,'01/05/2022 12:00:00 AM',0)
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Exemple 5 – Scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, il a été identifié que suite à une erreur d'équipement, des produits fabriqués le 5 janvier étaient défectueux. L'utilisateur final souhaite un objet de graphique qui affiche, par date, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués le 5 janvier.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données chargé dans une table appelée 'Products' (Produits).
- La table contient les champs suivants :
    - ID de produit
    - Heure de fabrication
    - Coût

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
=dayname(manufacture_date)
```

Créez les mesures suivantes :

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/01/2022	Faultless	44.67
01/02/2022	Faultless	36.34
01/03/2022	Faultless	51.75
01/04/2022	Faultless	89.69
01/05/2022	Defective	170.78
01/06/2022	Faultless	74.23

La fonction `inday()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué le 5 janvier, la fonction `inday()` renvoie une valeur booléenne égal à TRUE et identifie les produits comme ‘Defective’ (défectueux). Tout produit renvoyant une valeur égale à FALSE, et par conséquent non fabriqué le jour en question, est identifié comme ‘Faultless’ (correct).

indaytotime

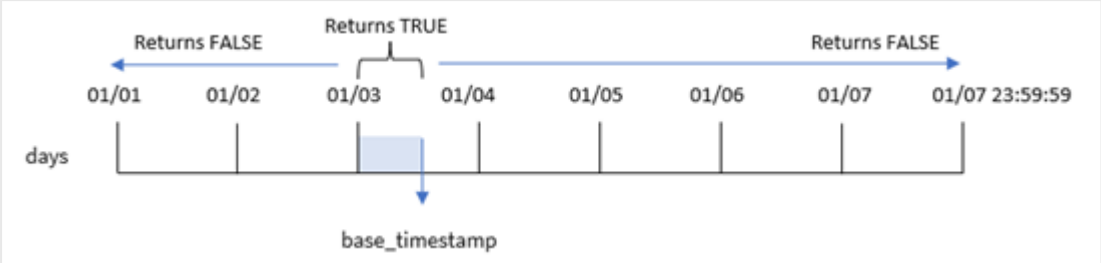
Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie du jour comprenant l'argument **base_timestamp** jusqu'à la milliseconde exacte spécifiée dans **base_timestamp**.

Syntaxe :

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

La fonction `indaytotime()` renvoie un résultat booléen suivant le moment auquel la valeur d'horodatage se produit au cours du segment du jour. La limite de début de ce segment est le début de la journée, qui est défini par défaut sur minuit ; il est possible de modifier le début de la journée via l'argument `day_start` de la fonction `indaytotime()`. La limite de fin du segment de la journée est déterminée par un argument `base_timestamp` de la fonction.

Diagramme de la fonction `indaytotime`.



Cas d'utilisation

La fonction `indaytotime()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. La fonction `indaytotime()` renvoie une agrégation ou un calcul suivant qu'un horodatage s'est produit au cours du segment de la journée jusqu'à l'heure, comprise, de l'horodatage de base.

Par exemple, la fonction `indaytotime()` peut-être utilisée pour indiquer la somme des ventes de billets de séances de cinéma qui ont eu lieu jusqu'à présent à la date d'aujourd'hui.

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Arguments

Argument	Description
`timestamp`	Date et heure à comparer à `base_timestamp`.
`base_timestamp`	Date et heure utilisées pour évaluer l'horodatage.
`period_no`	Il est possible de décaler le jour à l'aide de l'argument `period_no`. `period_no` est un entier, où la valeur 0 indique le jour comprenant l'argument `base_timestamp`. Les valeurs négatives de l'argument `period_no` indiquent les jours passés, tandis que les valeurs positives désignent les jours à venir.
`day_start`	(facultatif) Si vous souhaitez utiliser des jours qui ne commencent pas à minuit, indiquez un décalage sous forme de fraction d'un jour dans `day_start`. Par exemple, utilisez 0.125 pour indiquer 3 h du matin.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Renvoie True
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Renvoie False
`indaytotime` '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Renvoie True

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour la période comprise entre le 4 et le 5 janvier est chargé dans une table appelée 'Transactions'.
- Un champ date fourni dans la variable système `TimeStamp` au format `(M/D/YYYY h:mm:ss[.fff] TT)`.
- Un chargement précédent contenant la fonction `indaytotime()` définie comme `'in_day_to_time'`, champ qui détermine si chacune des transactions a lieu avant 9h du matin.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM',0) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
```

```
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Résultats

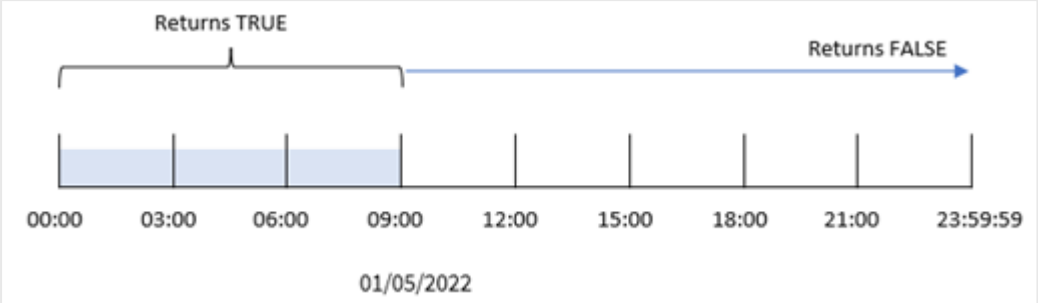
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- in_day_to_time

Tableau de résultats

date	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exemple 1 - Diagramme de la fonction `indaytotime` avec une limite de 9h00 du matin



Le champ `in_day_to_time field` est créé dans l'instruction LOAD précédente via la fonction `indaytotime ()` et en transmettant le champ date, un horodatage codé en dur pour le 5 janvier à 9h00 et un décalage égal à 0 comme arguments de la fonction. Toute transaction ayant lieu entre minuit et 9h00 du matin le 5 janvier renvoie la valeur TRUE.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le script de chargement utilise le même ensemble de données et le même scénario que ceux utilisés dans le premier exemple.

Cependant, dans cet exemple, vous allez calculer si la date de la transaction a eu lieu un jour avant 9h00 le 5 janvier.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', -1) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
```

```
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Résultats

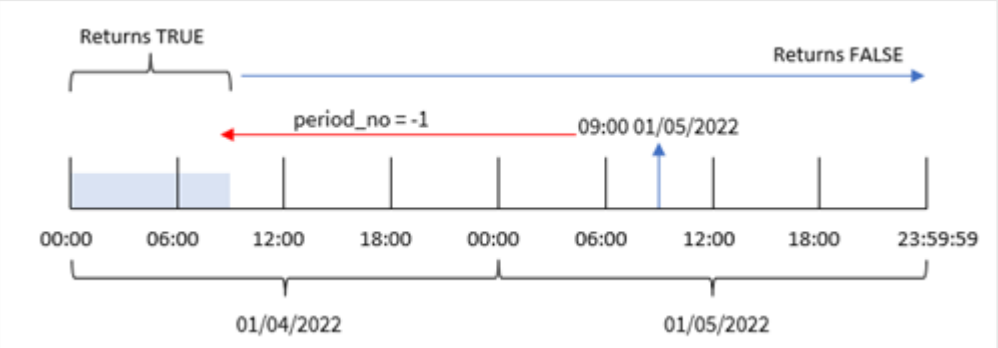
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_day_to_time`

Tableau de résultats

date	in_day_to_time
01/04/2022 3:41:54 AM	-1
01/04/2022 4:19:43 AM	-1
01/04/2022 04:53:47 AM	-1
01/04/2022 8:38:53 AM	-1
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	0
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exemple 2 - Diagramme de la fonction `indaytotime` avec des transactions à partir du 4 janvier.



Dans cet exemple, étant donné qu'un décalage de -1 a été utilisé comme argument de décalage dans la fonction `indaytotime()`, la fonction détermine si chaque date de transaction a eu lieu avant 9h00 le 4 janvier. Cela peut être vérifié dans le tableau de résultats, dans lequel une transaction renvoie un résultat booléen égal à TRUE.

Exemple 3 – day_start

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la politique de l'entreprise détermine que la journée de travail commence et se termine à 8h00 du matin.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', 0,8/24) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
```

```
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Résultats

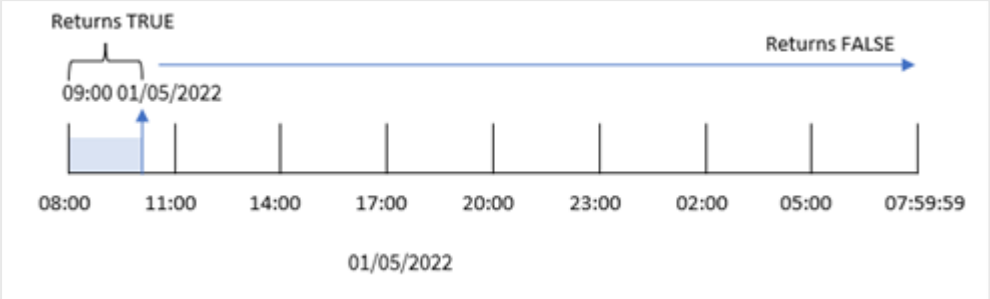
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_day_to_time`

Tableau de résultats

date	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exemple 3 - Diagramme de la fonction `indaytotime` avec des transactions de 8h00 à 9h00 du matin.



Étant donné que l'argument `start_day` de 8/24, qui équivaut à 8h du matin, est utilisé dans la fonction `indaytotime()`, chaque journée commence et se termine à 8h du matin. Par conséquent, la fonction `indaytotime()` renverra un résultat booléen égal à TRUE pour tout transaction ayant eu lieu entre 8h du matin et 9h du matin le 5 janvier.

Exemple 4 – Objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Vous allez effectuer un calcul pour déterminer si une transaction a lieu le 5 janvier avant 9h00 du matin en créant une mesure dans un objet graphique.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

`date`.

Pour déterminer si une transaction a lieu le 5 janvier avant 9h00 du matin, créez la mesure suivante :

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Tableau de résultats

date	=indaytotime(date,'01/05/2022 9:00:00 AM',0)
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

La mesure `in_day_to_time` est créée dans l'objet de graphique via la fonction `indaytotime()` et en transmettant le champ date, un horodatage codé en dur pour le 5 janvier à 9h00 et un décalage égal à `0` comme arguments de la fonction. Toute transaction ayant lieu entre minuit et 9h00 du matin le 5 janvier renvoie la valeur TRUE. Cela est validé dans le tableau de résultats.

Exemple 5 – Scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, un ensemble de données contenant les ventes de billets d'un cinéma local est chargé dans une table appelée Ticket_Sales. Aujourd'hui, nous sommes le 3 mai 2022 et il est 11h du matin.

L'utilisateur souhaite un objet de graphique KPI indiquant les revenus obtenus de toutes les séances qui ont eu lieu jusqu'à présent aujourd'hui.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Ticket_Sales:
Load
*
Inline
[
sale ID, show time, ticket price
1,05/01/2022 09:30:00 AM,10.50
2,05/03/2022 05:30:00 PM,21.00
3,05/03/2022 09:30:00 AM,10.50
4,05/03/2022 09:30:00 AM,31.50
5,05/03/2022 09:30:00 AM,10.50
6,05/03/2022 12:00:00 PM,42.00
7,05/03/2022 12:00:00 PM,10.50
8,05/03/2022 05:30:00 PM,42.00
9,05/03/2022 08:00:00 PM,31.50
10,05/04/2022 10:30:00 AM,31.50
11,05/04/2022 12:00:00 PM,10.50
12,05/04/2022 05:30:00 PM,10.50
13,05/05/2022 05:30:00 PM,21.00
14,05/06/2022 12:00:00 PM,21.00
15,05/07/2022 09:30:00 AM,42.00
16,05/07/2022 10:30:00 AM,42.00
17,05/07/2022 10:30:00 AM,10.50
18,05/07/2022 05:30:00 PM,10.50
19,05/08/2022 05:30:00 PM,21.00
20,05/11/2022 09:30:00 AM,10.50
];
```

Résultats

Procédez comme suit :

1. Créez un objet KPI.
2. Créez une mesure qui indiquera la somme de toutes les ventes de billets des séances qui ont eu lieu aujourd'hui jusqu'à présent via la fonction `indaytotime()` :

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Créez une étiquette pour l'objet KPI, ‘Current Revenue’.
4. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

La somme totale des ventes de billets jusqu'à 11h du matin le 3 mai 2022 est égale à 52,50 $.

La fonction `indaytotime ()` renvoie une valeur booléenne lors de la comparaison des heures des séances de chaque vente de billet à l'heure actuelle ('05/03/2022 11:00:00 AM'). Pour toute séance ayant eu lieu le 3 mai avant 11h du matin, la fonction `indaytotime ()` renvoie une valeur booléenne égale à TRUE et le prix de son billet sera inclus dans la somme totale.

inlunarweek

Cette fonction détermine si l'argument **timestamp** tombe pendant la semaine lunaire comprenant l'argument **base_date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine. À l'exception de la dernière semaine de l'année, chaque semaine contiendra exactement sept jours.

Syntaxe :

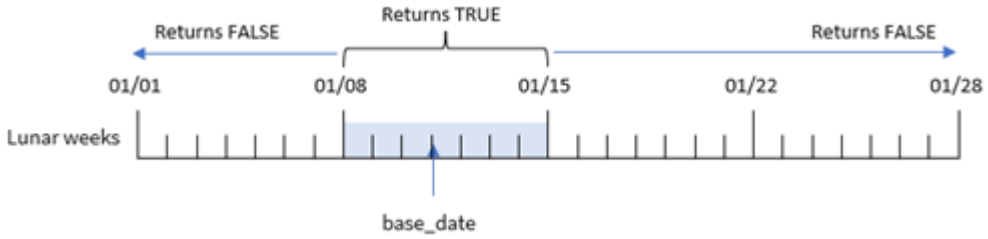
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Type de données renvoyé : booléen

> *Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.*

La fonction `inlunarweek()` détermine la semaine lunaire de la `base_date`. Elle renvoie ensuite un résultat booléen une qu'elle a déterminé si chaque valeur d'horodatage tombe pendant la même semaine lunaire que la `base_date`.

Diagramme de la fonction `inlunarweek()`



Cas d'utilisation

La fonction `inlunarweek()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression IF. Cela renverrait une agrégation ou un calcul suivant que la date évaluée s'est produite ou non au cours de la semaine lunaire en question.

Par exemple, la fonction `inlunarweek()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'une semaine lunaire donnée.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.

Argument	Description
base_date	Date utilisée pour évaluer la semaine lunaire.
period_no	Il est possible de décaler la semaine lunaire à l'aide de l'argument **period_no**. period_no est un entier, où la valeur 0 indique la semaine lunaire comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines lunaires passées tandis que les valeurs positives désignent les semaines lunaires à venir.
first_week_day	Décalage pouvant être supérieur ou inférieur à zéro. Il modifie le début de l'année du nombre de jours et/ou de fractions de jour spécifié.

Exemples de fonction

Exemple	Résultat
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Renvoie `TRUE`, car la valeur de `timestamp`, 01/12/2013, tombe pendant la semaine du 01/08/2013 au 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Renvoie `FALSE`, car la `base_date` 01/07/2013 tombe pendant la semaine lunaire définie du 01/01/2013 au 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Renvoie `FALSE`. La définition de la valeur de period_no sur -1 décale la semaine à la semaine précédente, du 01/01/2013 au 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Renvoie `TRUE`. Par rapport à l'exemple précédent, l'`timestamp` tombe pendant la semaine suivante, après avoir pris en compte le décalage en arrière.
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Renvoie `FALSE`. La définition d'une valeur 3 pour `first_week_day` signifie que le début de l'année est calculé à partir du 01/04/2013. Par conséquent, la valeur de `base_`date tombe pendant la première semaine et la valeur de `timestamp` tombe pendant la semaine du 01/11/2013 au 01/17/2013.

La fonction `inlunarweek()` est souvent utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
lunarweekname (page 860)	Cette fonction permet de déterminer le nombre de semaines lunaires de l'année pendant lesquelles tombe une date d'entrée.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de transactions pour le mois de janvier, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

Créez un champ, `in_lunar_week`, qui détermine si les transactions ont eu lieu pendant la même semaine lunaire que le 10 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
```

```
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Résultats

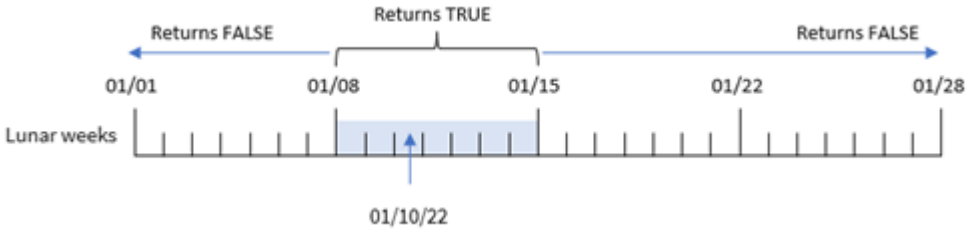
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_lunar_week`

Tableau de résultats

date	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Fonction `inlunarweek()`, exemple de base



Le champ `in_lunar_week` est créé dans l'instruction LOAD précédente via la fonction `inlunarweek()`, puis en transmettant les éléments suivants comme arguments de la fonction :

- Le champ `date`
- Une date codée en dur pour le 10 janvier comme `base_date`
- Un argument `period_no` égal à 0

Étant donné que les semaines lunaires commencent le 1er janvier, le 10 janvier tomberait pendant la semaine lunaire qui commence le 8 janvier et qui se termine le 14 janvier. Par conséquent, toutes les transactions qui se produisent entre ces deux dates de janvier renverraient une valeur booléenne `TRUE`. Cela est validé dans le tableau de résultats.

Exemple 2 - period_no

Exemples et résultats :

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

Cependant, dans cet exemple, la tâche consiste à créer un champ, `2_lunar_weeks_later`, qui détermine si les transactions ont eu lieu ou non deux semaines lunaires après le 10 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
```

```
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

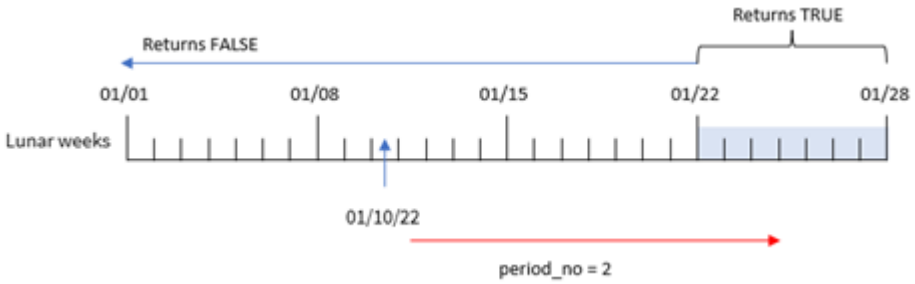
- `date`
- `2_lunar_weeks_later`

Tableau de résultats

date	2_lunar_weeks_later
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0
1/9/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/13/2022	0
1/14/2022	0
1/15/2022	0

date	2_lunar_weeks_later
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	-1
1/23/2022	-1

Fonction `inlunarweek()`, exemple period_no



Dans cet exemple, étant donné qu'un argument `period_no` égal à 2 a été utilisé comme argument de décalage dans la fonction `inlunarweek()`, la fonction définit la semaine commençant le 22 janvier comme la semaine lunaire par rapport à laquelle valider les transactions. Par conséquent, toute transaction effectuée entre le 22 janvier et le 28 janvier renverra un résultat booléen `TRUE`.

Exemple 3 - first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement utilise le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans l'exemple, nous définissons les semaines lunaires de sorte qu'elles commencent le 6 janvier.

- Même ensemble de données et même scénario que ceux du premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- Argument `first_week_day` égal à 5. Cela détermine le début des semaines lunaires le 5 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,5) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

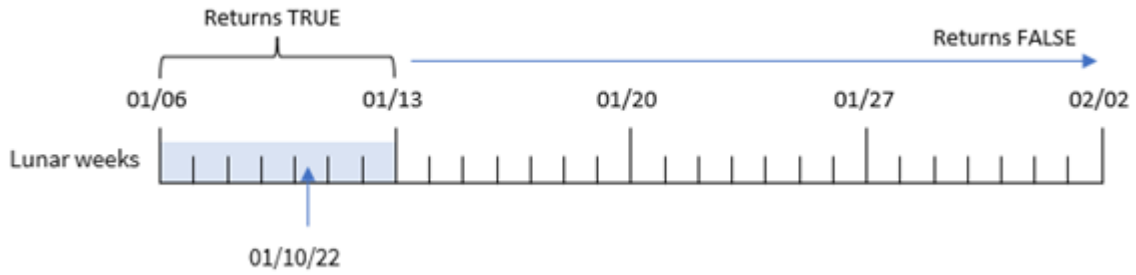
- `date`
- `in_lunar_week`

Tableau de résultats

date	in_lunar_week
1/5/2022	0
1/6/2022	-1
1/7/2022	-1
1/8/2022	-1

date	in_lunar_week
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Fonction `inlunarweek()`, exemple first_week_day



Dans cet exemple, étant donné que l'argument `first_week_date` égal à 5 est utilisé dans la fonction `inlunarweek()`, il décale le début du calendrier de semaines lunaires au 6 janvier. Par conséquent, le 10 janvier tombe pendant la semaine lunaire commençant le 6 janvier et se terminant le 12 janvier. Toute transaction qui tombe entre ces deux dates renverra une valeur booléenne `TRUE`.

Exemple 4 - objet graphique

Script de chargement et expression de graphique :

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu pendant la même semaine lunaire que celle du 10 janvier est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer si une transaction a lieu pendant la semaine lunaire contenant le 10 janvier, créez la mesure suivante :

```
= inlunarweek(date,'01/10/2022', 0)
```

Tableau de résultats

date	=inlunarweek(date,'01/10/2022', 0)
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Exemple 5 - scénario

Script de chargement et expression de graphique :

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Products`.
- Informations incluant l'ID du produit, la date de fabrication et le prix de revient.

Il a été identifié que, suite à une erreur d'équipement, des produits fabriqués au cours de la semaine lunaire incluant le 12 janvier étaient défectueux. L'utilisateur final souhaite un objet graphique qui affiche, par nom de semaine lunaire, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués ce mois-là.

Script de chargement

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

Transactions:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez une dimension pour afficher les noms de mois :
   `=lunarweekname(manufacture_date)`
3. Créez une mesure pour identifier les produits défectueux et les produits corrects via la fonction `inlunarweek()` :
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Créez une mesure pour additionner les valeurs `cost_price` des produits :

```
=sum(cost_price)
```

5. Définissez le **Formatage des nombres** des mesures sur **Devise**.
6. Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

lunarweekname (manufacture_date)	=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	sum(cost_ price)
2022/01	Faultless	$125.79
2022/02	Defective	$316.38
2022/03	Faultless	$455.75
2022/04	Faultless	$146.09

La fonction `inlunarweek()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué au cours de la semaine lunaire qui contient le 10 janvier, la fonction `inlunarweek()` renvoie une valeur booléenne `TRUE` et identifie les produits comme 'Defective' (défectueux). Tout produit renvoyant une valeur `FALSE`, et par conséquent non fabriqué au cours de cette semaine-là, est identifié comme 'Faultless' (correct).

inlunarweektodate

Cette fonction détermine si l'argument **timestamp** se trouve dans la partie de la semaine lunaire jusqu'à la dernière milliseconde spécifiée dans **base_date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

Syntaxe :

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Exemple de diagramme de la fonction `inlunarweektodate()`

La fonction `inlunarweektodate()` joue le rôle de point final de la semaine lunaire. En revanche, la fonction `inlunarweek()` détermine la semaine lunaire de la `base_date`. Par exemple, si la `base_date` était le 5 janvier, tout horodatage entre le 1er janvier et le 5 janvier renverrait un résultat booléen `TRUE`, tandis que les dates du 6 et du 7 janvier et les dates ultérieures renverraient un résultat booléen `FALSE`.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer la semaine lunaire.
period_no	Il est possible de décaler la semaine lunaire à l'aide de l'argument **period_no**. period_no est un entier, où la valeur 0 indique la semaine lunaire comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines lunaires passées tandis que les valeurs positives désignent les semaines lunaires à venir.
first_week_day	Décalage pouvant être supérieur ou inférieur à zéro. Il modifie le début de l'année du nombre de jours et/ou de fractions de jour spécifié.

Cas d'utilisation

La fonction `inlunarweektodate()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression IF. La fonction `inlunarweektodate()` serait utilisée lorsque l'utilisateur souhaite que le calcul renvoie une agrégation ou un calcul, suivant que la date évaluée a eu lieu ou non pendant un segment donné de la semaine en question.

Par exemple, la fonction `inlunarweektodate()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'une semaine donnée jusqu'à une date spécifique incluse.

Exemples de fonction

Exemple	Résultat
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Renvoie `TRUE`, car la valeur de l'`timestamp`, 01/12/2013, tombe pendant la partie de la semaine du 01/08/2013 au 01/13/2013.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Renvoie `FALSE`, car la valeur de l'`timestamp` est postérieure à la valeur de la `base_date`, même si les deux dates font partie de la même semaine lunaire précédant le 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Renvoie `TRUE`. La définition de la valeur 1 pour `period_no` a pour effet de retarder la date de référence `base_date` d'une semaine. De ce fait, la valeur de `timestamp` tombe dans la partie de la semaine lunaire.

La fonction `inlunarweektodate()` est souvent utilisée en combinaison avec les fonctions suivantes :

Fonctions associées

Fonction	Interaction
lunarweekname (page 860)	Cette fonction permet de déterminer le nombre de semaines lunaires de l'année pendant lesquelles tombe une date d'entrée.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour le mois de janvier, chargé dans une table appelée `Transactions`. La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- Créez un champ, `in_lunar_week_to_date`, qui détermine les transactions qui ont eu lieu pendant la semaine lunaire jusqu'à la date du 10 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 0) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
```

```
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_lunar_week_to_date`

Tableau de résultats

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Fonction `inlunarweektodate()`, aucun argument supplémentaire

Le champ `in_lunar_week_to_date` est créé dans l'instruction LOAD précédente via la fonction `inlunarweektodate()` et en transmettant le champ `date`, une date codée en dur pour le 10 janvier comme notre `base_date` et un décalage égal à 0 comme arguments de la fonction.

Étant donné que les semaines lunaires commencent le 1er janvier, le 10 janvier tomberait pendant la semaine lunaire qui commence le 8 janvier ; et, parce que nous utilisons la fonction `inlunarweektodate()`, cette semaine lunaire se terminerait donc le 10. Par conséquent, toutes les transactions qui se produisent entre ces deux dates de janvier renverraient une valeur booléenne `TRUE`. Cela est validé dans le tableau de résultats.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, la tâche consiste à créer un champ, `2_lunar_weeks_later`, qui détermine si les transactions ont eu lieu ou non deux semaines après la semaine lunaire jusqu'à la date du 1er janvier.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
```

```
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Résultats

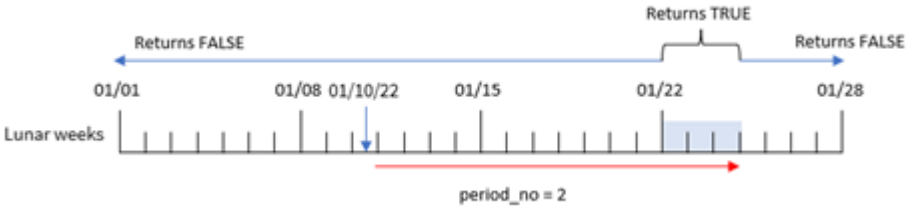
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `2_lunar_weeks_later`

Tableau de résultats

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Fonction `inlunarweektodate()`, exemple period_no



Dans cet exemple, la fonction `inlunarweektodate()` détermine que la semaine lunaire jusqu'au 10 janvier compte trois jours (les 8, 9 et 10 janvier). Étant donné qu'un argument `period_no` égal à 2 a été utilisé comme argument de décalage, cette semaine lunaire est décalée de 14 jours. Par conséquent, cette semaine lunaire de trois jours inclut les 22, 23 et 24 janvier. Toute transaction effectuée entre le 22 janvier et le 24 janvier renverra un résultat booléen `TRUE`.

Exemple 3 - first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- La variable système `DateFormat` par défaut MM/DD/YYYY est utilisée.
- Argument `first_week_date` égal à 3. Cela détermine le début des semaines lunaires le 3 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,3) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
```

```
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_lunar_week_to_date`

Tableau de résultats

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	-1
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Fonction `inlunarweektodate()`, exemple first_week_day

Dans cet exemple, étant donné que l'argument `the first_week_date` égal à 3 est utilisé dans la fonction `inlunarweek()`, la première semaine lunaire se déroulera du 3 janvier au 10 janvier. Étant donné que le 10 janvier est également la `base_date`, toute transaction qui tombe entre ces deux dates renverra une valeur booléenne `TRUE`.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu pendant la semaine lunaire jusqu'au 10 janvier est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
```

```
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

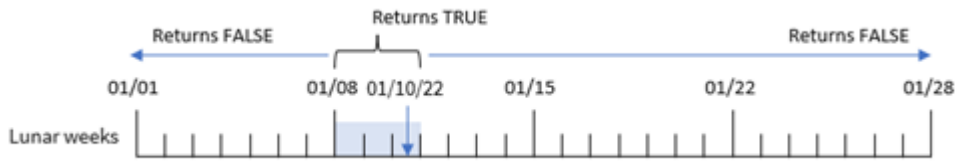
Créez la mesure suivante :

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Tableau des résultats

date	=inlunarweektodate(date,'01/10/2022', 0)
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Fonction `inlunarweektodate()`, exemple objet graphique



La mesure `in_lunar_week_to_date` est créée dans l'objet graphique via la fonction `inlunarweektodate()` et en transmettant le champ date, une date codée en dur pour le 10 janvier comme notre `base_date` et un décalage égal à 0 comme arguments de la fonction.

Étant donné que les semaines lunaires commencent le 1er janvier, le 10 janvier tomberait pendant la semaine lunaire qui commence le 8 janvier . De plus, parce que nous utilisons la fonction `inlunarweektodate()`, cette semaine lunaire se terminerait le 10. Par conséquent, toutes les transactions qui se produisent entre ces deux dates de janvier renverraient une valeur booléenne `TRUE`. Cela est validé dans le tableau de résultats.

Exemple 5 - scénario

Script de chargement et expressions de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Products`.
- Informations incluant l'ID du produit, la date de fabrication et le prix de revient.

Il a été identifié que, suite à une erreur d'équipement, des produits fabriqués pendant la semaine lunaire du 12 janvier étaient défectueux. Le problème a été résolu le 13 janvier. L'utilisateur final souhaite un objet graphique qui affiche, par semaine, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués cette semaine-là.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022 12:22:06',37.23
8189,'01/05/2022 01:02:30',17.17
8190,'01/06/2022 15:36:20',88.27
8191,'01/08/2022 10:58:35',57.42
8192,'01/09/2022 08:53:32',53.80
8193,'01/10/2022 21:13:01',82.06
8194,'01/11/2022 00:57:13',40.39
8195,'01/12/2022 09:26:02',87.21
8196,'01/13/2022 15:05:09',95.93
8197,'01/14/2022 18:44:57',45.89
8198,'01/15/2022 06:10:46',36.23
8199,'01/16/2022 06:39:27',25.66
```

```
8200,'01/17/2022 10:44:16',82.77
8201,'01/18/2022 18:48:17',69.98
8202,'01/26/2022 04:36:03',76.11
8203,'01/27/2022 08:07:49',25.12
8204,'01/28/2022 12:24:29',46.23
8205,'01/30/2022 11:56:56',84.21
8206,'01/30/2022 14:40:19',96.24
8207,'01/31/2022 05:28:21',67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez une dimension pour afficher les noms de semaine :
   `=weekname(manufacture_date)`
3. Ensuite, créez une dimension qui utilise la fonction `inlunarweektodate()` pour identifier les produits défectueux et les produits corrects :
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Créez une mesure pour additionner les valeurs `cost_price` des produits :
   `=sum(cost_price)`
5. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	=Sum(cost_ price)
2022/01	Faultless	$142.67
2022/02	Defective	$320.88
2022/02	Faultless	$141.82
2022/03	Faultless	$214.64
2022/04	Faultless	$147.46
2022/05	Faultless	$248.12

La fonction `inlunarweektodate()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Les produits qui renvoient une valeur booléenne `TRUE` sont identifiés comme `'Defective'`. Tout produit renvoyant une valeur `FALSE` et par conséquent non fabriqué au cours de la semaine lunaire jusqu'au 12 janvier est identifié comme `'Faultless'`.

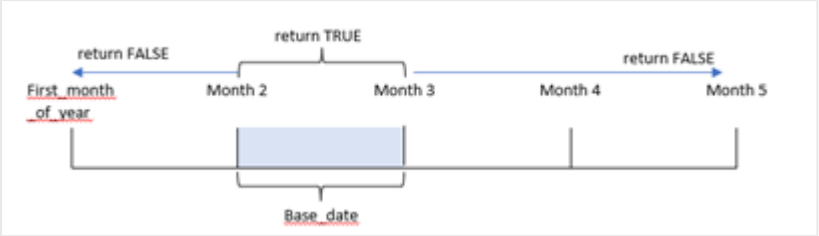
inmonth

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le mois comprenant l'argument **base_date**.

Syntaxe :

```
InMonth (timestamp, base_date, period_no)
```

Diagramme de la fonction `indaytotime`.



En d'autres termes, la fonction `inmonth()` détermine si un ensemble de dates tombe pendant ce mois et renvoie une valeur booléenne basée sur une valeur `base_date` qui identifie le mois.

Cas d'utilisation

La fonction `inmonth()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. Cela renvoie une agrégation ou un calcul, suivant qu'une date s'est produite ou non au cours du mois, y compris la date en question.

Par exemple, la fonction `inmonth()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'un mois spécifique.

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Arguments

Argument	Description
timestamp	Date à comparer à `base_date`.
base_date	Date utilisée pour évaluer le mois. Il est important de noter que l'argument `base_date` peut être n'importe quel jour d'un mois.
period_no	Il est possible de décaler le mois à l'aide de l'argument `period_no`. `period_no` est un entier, où la valeur 0 indique le mois comprenant l'argument `base_date`. Les valeurs négatives spécifiées pour `period_no` indiquent les mois passés, tandis que les valeurs positives désignent les mois à venir.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inmonth ('25/01/2013', '01/01/2013', 0)`	Renvoie True
`inmonth('25/01/2013', '23/04/2013', 0)`	Renvoie False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Renvoie False
`inmonth ('25/12/2012', '17/01/2013', -1)`	Renvoie True

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour le premier semestre 2022.
- Chargement précédent avec une variable supplémentaire, '`in_month`', qui détermine si les transactions ont eu lieu en avril.

Script de chargement

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

Transactions:
  Load
    *,
    inmonth(date,'04/01/2022', 0) as in_month
    ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
```

```
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_month`

Exemples de fonction

date	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Le champ '`in_month`' est créé dans l'instruction LOAD précédente via la fonction `inmonth()` et en transmettant le champ date, une date codée en dur du 1 avril, comme `base_date`, et une valeur `period_no` égale à `0`, comme arguments de la fonction.

L'argument `base_date` identifie le mois qui renverra un résultat booléen TRUE. Par conséquent, toutes les transactions survenues en avril renvoient TRUE, valeur validée dans le tableau de résultats.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, vous allez créer un champ, '2_months_prior', qui détermine si les transactions ont eu lieu deux mois avant avril.

Script de chargement

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

Transactions:
Load
   *,
   inmonth(date,'04/01/2022', -2) as [2_months_prior]
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- 2_months_prior

Exemples de fonction

date	2_months_prior
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0

date	2_months_prior
2/1/2022	-1
2/2/2022	-1
2/20/2022	-1
4/11/2022	0
4/13/2022	0
4/15/2022	0
4/25/2022	0
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

L'utilisation de -2 comme argument `period_no` dans la fonction `inmonth()` décale le mois défini par l'argument `base_date` deux mois avant. Dans cet exemple, le mois défini d'avril devient février.

Par conséquent, toute transaction effectuée en février renverra un résultat booléen TRUE.

Exemple 3 – objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux des précédents exemples sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu en avril est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
```

```
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Objet graphique

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
date
```

Pour calculer si une transaction a lieu en avril, créez la mesure suivante :

```
=inmonth(date,'04/01/2022', 0)
```

Résultats

	Exemples de fonction
date	**=inmonth(date,'04/01/2022', 0)**
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Exemple 4 – scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Products`'. La table contient les champs suivants :

- Product ID
- Manufacture date
- Cost price

En raison d'une erreur d'équipement, les produits fabriqués en juillet 2022 étaient défectueux. Le problème a été résolu le 27 juillet 2022.

L'utilisateur final souhaite un graphique qui affiche, par mois, l'état des produits fabriqués 'defective' (défectueux) (valeur booléenne TRUE) ou 'faultless' (corrects) (valeur booléenne FALSE) et le coût des produits fabriqués au cours de ce mois-là.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
=monthname(manufacture_date)
```

Créez les mesures suivantes :

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).
2. Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69
Mar 2022	Faultless	$53.80
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$214.64
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

La fonction `inmonth()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué en juillet 2022, la fonction `inmonth()` renvoie une valeur booléenne TRUE et identifie les produits comme 'Defective' (défectueux). Tout produit renvoyant une valeur FALSE, et par conséquent non fabriqué en juillet, est identifié comme 'Faultless' (correct).

inmonths

Cette fonction permet de déterminer si un horodatage tombe pendant la même période (mois, période de deux mois, trimestre, période de quatre mois ou semestre) que la date de référence. Il est également possible de déterminer si l'horodatage se situe dans une période passée ou future.

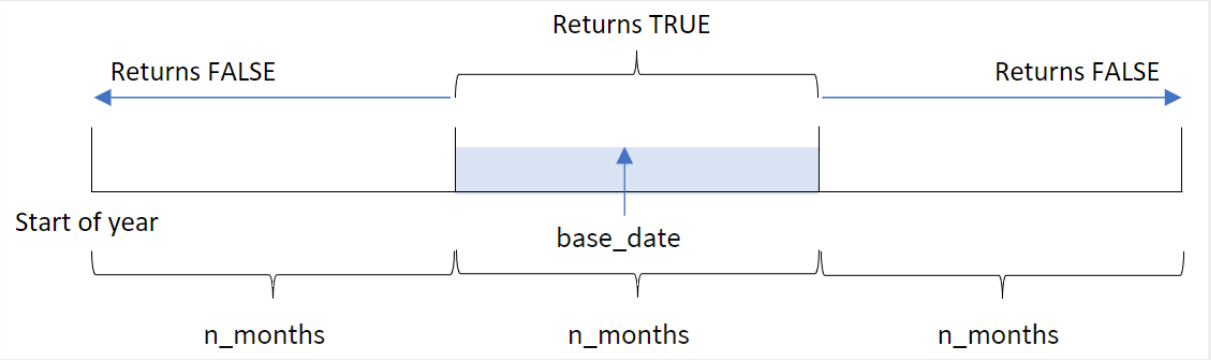
Syntaxe :

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Diagramme de la fonction `inmonths()`



La fonction `inmonths()` divise l'année en segments en fonction de l'argument `n_months` fourni. Elle détermine ensuite si chaque horodatage évalué tombe dans le même segment que celui de l'argument `base_date`. Si, toutefois, un argument `period_no` est fourni, la fonction détermine si les horodatages tombent pendant une période précédente ou suivante par rapport à la `base_date`.

Les segments suivants de l'année sont disponibles dans la fonction en tant qu'arguments `n_month`.

Arguments `n_month`

Période	Nombre de mois
mois	1
période de deux mois	2
trimestre	3
période de quatre mois	4
semestre	6

Cas d'utilisation

La fonction `inmonths()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. En utilisant la fonction `inmonths()`, vous pouvez sélectionner la période à évaluer. Par exemple, en laissant l'utilisateur identifier les produits fabriqués au cours du mois, du trimestre ou du semestre d'une période donnée.

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Arguments

Argument	Description
n_months	Nombre de mois définissant la période. Entier ou expression aboutissant à un entier qui doit correspondre à l'une de ces valeurs : 1 (qui équivaut à la fonction inmonth()), 2 (période de deux mois), 3 (qui équivaut à la fonction inquarter()), 4 (période de quatre mois) ou 6 (semestre).

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer la période.
period_no	Il est possible de décaler la période à l'aide de l'argument **period_no**, d'un entier ou d'une expression aboutissant à un entier, où la valeur 0 indique la période comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les périodes passées tandis que les valeurs positives désignent les périodes à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Renvoie TRUE. Car la valeur d'horodatage, 01/25/2013, est comprise dans la période de quatre mois du 01/01/2013 au 04/30/2013, qui comprend la valeur de `base_date`, 04/25/2013.
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Renvoie FALSE. Car la date 05/25/2013 se trouve en dehors de la période indiquée dans l'exemple précédent.
`inmonths(4, '11/25/2012', '02/01/2013', -1)`	Renvoie TRUE. Car la valeur de `period_no`, `-1`, décale la période de recherche de quatre mois en arrière (la valeur de n-months), ce qui définit la période de recherche du 09/01/2012 au 12/31/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Renvoie TRUE. Car la valeur de `first_month_of_year` est définie sur `3`, ce qui définit la période de recherche du 03/01/2006 au 07/30/2006 au lieu du 01/01/2006 au 04/30/2006.

Exemple 1 - aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Chargement précédent avec une variable supplémentaire, '`in_months`', qui détermine si les transactions ont eu lieu au cours du même trimestre que celui du 15 mai 2022.

Script de chargement

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

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0) as in_months
       ;
Load
*
```

```
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_months`

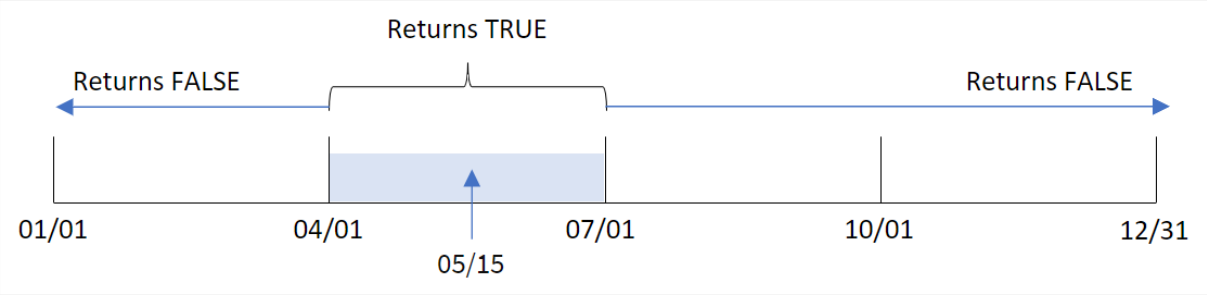
Tableau de résultats

date	in_months
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1

date	in_months
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ '`in_months`' est créé dans l'instruction LOAD précédente à l'aide de la fonction `inmonths()`. Le premier argument fourni est `3`, qui divise l'année en segments trimestriels. Le deuxième argument identifie le champ en cours d'évaluation, le champ date, dans cet exemple. Le troisième argument est une date codée en dur pour le 15 mai, qui est la `base_date`, et l'argument final est un argument `period_no` égal à `0`.

Diagramme de la fonction `inmonths()` avec des segments trimestriels



Mai tombe pendant le deuxième trimestre de l'année. Par conséquent, toute transaction effectuée entre le 1er avril et le 30 juin renverra un résultat booléen TRUE. Cela est validé dans le tableau de résultats.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée 'Transactions'.
- Chargement précédent avec une variable supplémentaire, 'previous_quarter', qui détermine si les transactions ont eu lieu au cours du trimestre avant le 15 mai 2022.

Script de chargement

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

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', -1) as previous_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

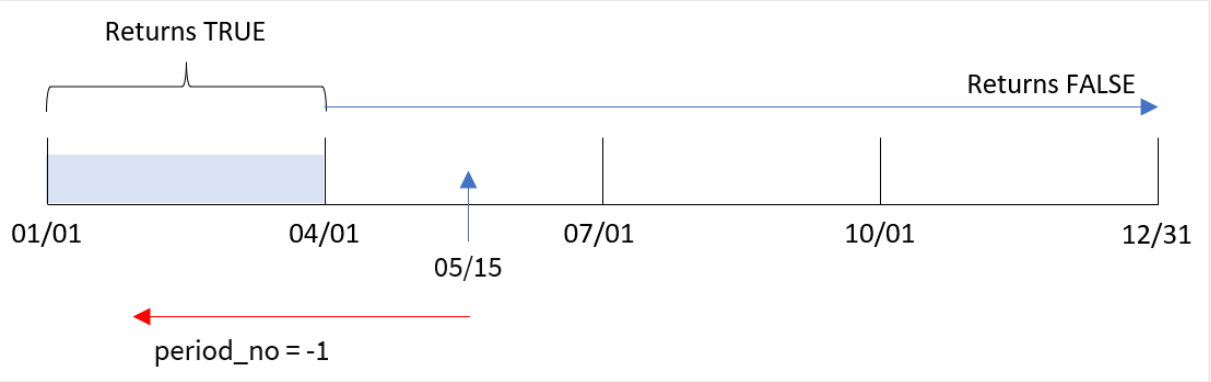
- date
- previous_quarter

Tableau de résultats

date	trimestre précédent
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La fonction évalue si les transactions ont eu lieu au cours du premier trimestre de l'année via la valeur `-1` comme argument `period_no` dans la fonction `inmonths()`. Le 15 mai est la `base_date` et tombe pendant le deuxième trimestre de l'année (avril-juin).

Diagramme de la fonction `inmonths()` avec des segments trimestriels et l'argument `period_no` défini sur `-1`



Par conséquent, toute transaction effectuée entre janvier et mars renverra un résultat booléen TRUE.

Exemple 3 - first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Chargement précédent avec une variable supplémentaire, '`in_months`', qui détermine si les transactions ont eu lieu au cours du même trimestre que celui du 15 mai 2022.

Dans cet exemple, la stratégie organisationnelle exige que mars soit le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0, 3) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
```

```
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_months`

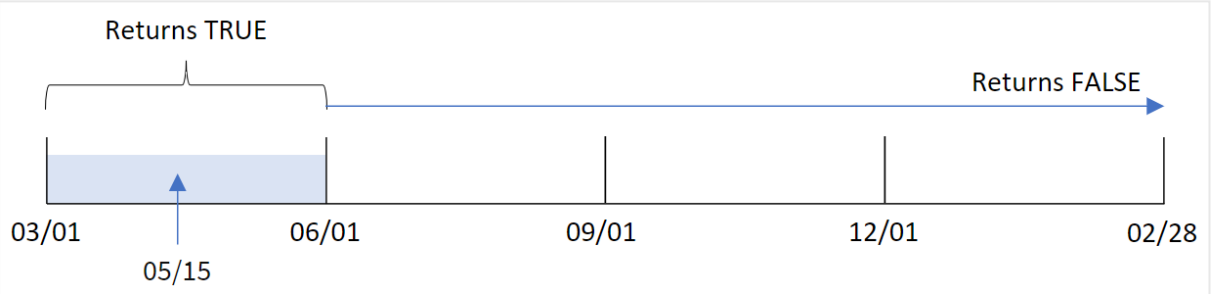
Tableau de résultats

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

date	in_months
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Si on utilise `3` comme argument `first_month_of_year` dans la fonction `inmonths()`, la fonction commence l'année le 1er mars. La fonction `inmonths()` divise ensuite l'année en trimestres : mars-mai, juin-août, sept-nov, déc-févr. Par conséquent, le 15 mai tombe pendant le premier trimestre de l'année (mars-mai).

Diagramme de la fonction `inmonths()` avec mars défini comme le premier mois de l'année.



Toute transaction qui a lieu au cours de ces mois renverra un résultat booléen TRUE.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu pendant le même trimestre que celui du 15 mai 2022 est créé sous forme de mesure dans un graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
```

```
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `date`

Pour calculer si des transactions ont eu lieu au cours du même trimestre que celui du 15 mai, créez la mesure suivante :

```
=inmonths(3,date,'05/15/2022', 0)
```

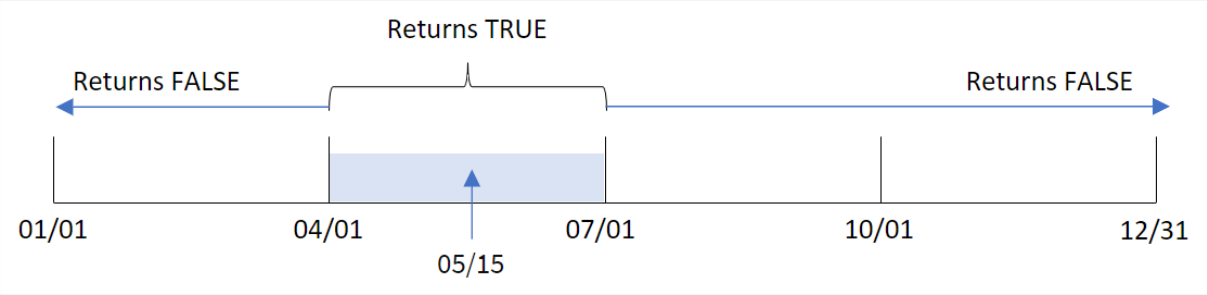
Tableau de résultats

date	=inmonths(3,date,'05/15/2022', 0)
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1

date	=inmonths(3,date,'05/15/2022', 0)
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ '`in_months`' est créé dans le graphique à l'aide de la fonction `inmonths()`. Le premier argument fourni est `3`, qui divise l'année en segments trimestriels. Le deuxième argument identifie le champ en cours d'évaluation, le champ date, dans cet exemple. Le troisième argument est une date codée en dur pour le 15 mai, qui est la `base_date`, et l'argument final est un argument `period_no` égal à `0`.

Diagramme de la fonction `inmonths()` avec des segments trimestriels



Mai tombe pendant le deuxième trimestre de l'année. Par conséquent, toute transaction effectuée entre le 1er avril et le 30 juin renverra un résultat booléen TRUE. Cela est validé dans le tableau de résultats.

Exemple 5 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée 'Products'.
- La table contient les champs suivants :
    - product ID
    - product type
    - manufacture date
    - cost price

L'utilisateur final souhaite un graphique qui affiche, par type de produit, le coût des produits fabriqués au cours du premier segment de 2021. L'utilisateur souhaite pouvoir définir la longueur de ce segment.

Script de chargement

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'2/19/2022',37.23
8189,product D,'3/7/2022',17.17
8190,product C,'3/30/2022',88.27
8191,product B,'4/5/2022',57.42
8192,product D,'4/16/2022',53.80
8193,product D,'5/1/2022',82.06
8194,product A,'5/7/2022',40.39
8195,product B,'5/22/2022',87.21
8196,product C,'6/15/2022',95.93
8197,product B,'6/26/2022',45.89
8198,product C,'7/9/2022',36.23
8199,product D,'7/22/2022',25.66
8200,product D,'7/23/2022',82.77
8201,product A,'7/27/2022',69.98
8202,product A,'8/2/2022',76.11
8203,product B,'8/8/2022',25.12
8204,product B,'8/19/2022',46.23
8205,product B,'9/26/2022',84.21
8206,product C,'10/14/2022',96.24
8207,product D,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille.

Au début du script de chargement, une variable, vPeriod, est créée ; elle est liée au contrôle d'entrée de variable.

Procédez comme suit :

1. Dans le panneau des ressources, cliquez sur **Objets personnalisés**.
2. Sélectionnez **Qlik Dashboard bundle** et créez un objet **Entrée de variable**.
3. Saisissez un titre pour l'objet graphique.
4. Sous **Variable**, sélectionnez **vPeriod** comme nom et définissez l'objet de sorte qu'il s'affiche sous forme de **Liste déroulante**.
5. Sous **Valeurs**, cliquez sur les valeurs **Dynamiques**. Saisissez les éléments suivants : `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Ajoutez une nouvelle table à la feuille.
7. Dans le panneau des propriétés, sous **Données**, ajoutez `product_type` comme dimension.
8. Ajoutez l'expression suivante comme mesure : `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Définissez le **Formatage des nombres** de la mesure sur **Money** (Devise).

Tableau de résultats

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
product A	$88.27
product B	$37.23
product C	$17.17
product D	$0.00

La fonction `inmonths()` utilise l'entrée utilisateur comme argument pour définir la taille du segment de début de l'année. La fonction transmet la date de fabrication de chacun des produits comme deuxième argument de la fonction `inmonths()`. Avec le 1er janvier utilisé comme troisième argument dans la fonction `inmonths()`, les produits dont les dates de fabrication tombent dans le segment de début de l'année renverront une valeur booléenne TRUE, et, par conséquent, la fonction sum additionnera les coûts de ces produits.

inmonthstodate

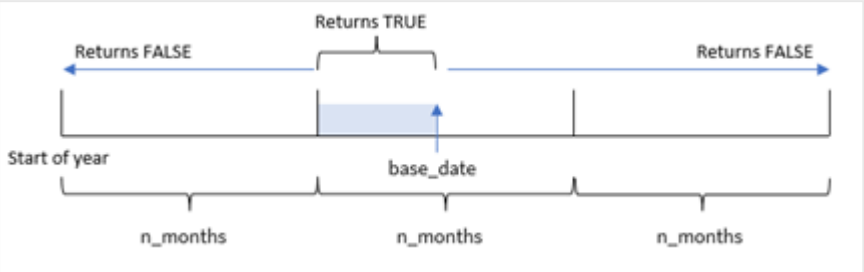
Cette fonction permet de déterminer si un horodatage tombe dans la partie d'une période (mois, période de deux mois, trimestre, période de quatre mois ou semestre) jusqu'à la dernière milliseconde incluse spécifiée dans `base_date`. Il est également possible de déterminer si l'horodatage se situe dans une période passée ou future.

Syntaxe :

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Type de données renvoyé : booléen

Diagramme de la fonction `inmonthstodate`.



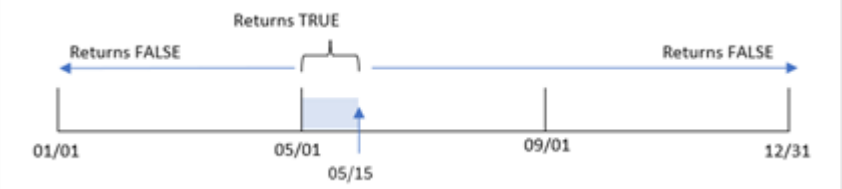
Arguments

Argument	Description
n_months	Nombre de mois définissant la période. Entier ou expression aboutissant à un entier qui doit correspondre à l'une de ces valeurs : 1 (qui équivaut à la fonction inmonth()), 2 (période de deux mois), 3 (qui équivaut à la fonction inquarter()), 4 (période de quatre mois) ou 6 (semestre).
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer la période.
period_no	Il est possible de décaler la période à l'aide de l'argument **period_no**, d'un entier ou d'une expression aboutissant à un entier, où la valeur 0 indique la période comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les périodes passées tandis que les valeurs positives désignent les périodes à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Dans la fonction `inmonthstodate()`, la `base_date` joue le rôle de point final du segment d'année donné dont elle fait partie.

Par exemple, si l'année était divisée en segments tertiaires et si la `base_date` était le 15 mai, tout horodatage entre le début du mois de janvier et la fin du mois d'avril renverrait un résultat booléen FALSE. Les dates comprises entre le 1er mai et le 15 mai renverraient TRUE. Le reste de l'année renverrait FALSE.

Diagramme de la plage de résultats booléens de la fonction `inmonthstodate`.



Les segments suivants de l'année sont disponibles dans la fonction en tant qu'arguments `n_month`.

Arguments `n_month`

Période	Nombre de mois
mois	1
période de deux mois	2
trimestre	3
tertile	4
semestre	6

Cas d'utilisation

La fonction `inmonthstodate()` renvoie un résultat booléen. Ce type de fonction est généralement utilisé comme condition dans une `if expression`. En utilisant la fonction `inmonthstodate()`, vous pouvez sélectionner la période à évaluer. Par exemple, en fournissant une variable d'entrée qui permet à l'utilisateur d'identifier les produits fabriqués au cours du mois, du trimestre ou du semestre d'une période, jusqu'à une certaine date.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Renvoie True, car la valeur de l'timestamp, 01/25/2013, est comprise dans la période de quatre mois du 01/01/2013 à la fin du 04/25/2013, qui comprend la valeur de la base_date, 04/25/2013.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Renvoie False, car le 04/26/2013 se trouve hors de la même période que celle indiquée dans l'exemple précédent.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Renvoie True, car la valeur de l'argument period_no, -1, décale la période de recherche d'une période de quatre mois en arrière (la valeur de n-months), ce qui définit la période de recherche du 01/09/2005 au 02/01/2006.

Exemple	Résultat
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Renvoie True, car la valeur de first_month_of_year est configurée sur 3, ce qui définit la période de recherche du 03/01/2006 au 06/01/2006 au lieu du 05/01/2006 au 06/01/2006.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée '`Transactions`'.
- Champ date dans la variable système `DateFormat` au format `(MM/DD/YYYY)`.
- Instruction LOAD précédente contenant les éléments suivants :
    - La fonction `inmonthstodate()` définie comme le champ '`in_months_to_date`'. Cela détermine les transactions qui ont eu lieu au cours du trimestre jusqu'au 15 mai 2022.

Script de chargement

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

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_months_to_date`

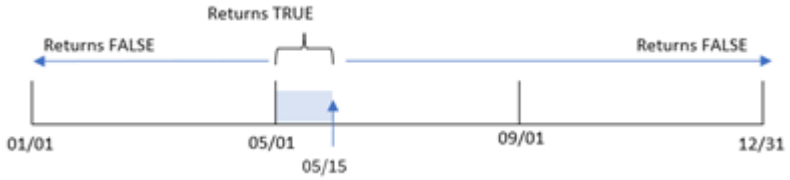
Tableau de résultats

date	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ 'in_months_to_date' est créé dans l'instruction LOAD précédente à l'aide de la fonction inmonthstodate().

Le premier argument fourni est 3, divisant l'année en segments trimestriels. Le deuxième argument identifie le champ en cours d'évaluation. Le troisième argument est une date codée en dur du 15 mai, qui est la base_date qui définit la limite de fin du segment. Le dernier argument est un argument period_no défini sur 0.

Diagramme de la fonction inmonthstodate sans argument supplémentaire.



Toute transaction qui se produit entre le 1er avril et le 15 mai renvoie un résultat booléen TRUE. Les dates des transactions en dehors de cette période renvoient FALSE.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la tâche consiste à créer un champ, 'previous_qtr_to_date', qui détermine si les transactions ont eu lieu un trimestre avant le 15 mai.

Script de chargement

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

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', -1) as previous_qtr_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
```

```
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_qtr_to_date`

Tableau de résultats

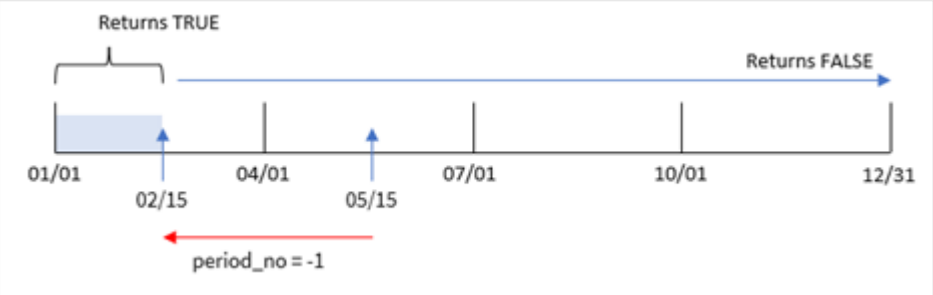
date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	previous_qtr_to_date
9/26/2022	0
10/14/2022	0
10/29/2022	0

Avec l'utilisation de `-1` comme argument `period_no` dans la fonction `inmonthstodate()`, la fonction décale les limites du segment d'année de comparaison d'un trimestre.

Le 15 mai tombe pendant le deuxième trimestre de l'année, et, par conséquent, le segment correspond initialement à la période entre le 1er avril et le 15 mai. L'argument `period_no` décale ce segment de trois mois négatifs. Les limites des dates deviennent du 1er janvier au 15 février.

Diagramme de la fonction `inmonthstodate` avec l'argument period_no défini sur -1.



Par conséquent, toute transaction effectuée entre le 1er janvier et le 15 février renverra un résultat booléen TRUE.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, la stratégie organisationnelle exige que mars soit le premier mois de l'exercice financier.

Créez un champ, '`in_months_to_date`', qui détermine les transactions qui ont eu lieu au cours du même trimestre jusqu'au 15 mai 2022.

Script de chargement

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

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0,3) as in_months_to_date
    ;
```

```
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_months_to_date`

Tableau de résultats

date	previous_qtr_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0

date	previous_qtr_to_date
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Si on utilise `3` comme argument `first_month_of_year` dans la fonction `inmonthstodate()`, la fonction commence l'année le 1er mars, puis la divise en trimestres en fonction du premier argument fourni. Par conséquent, les segments de trimestre sont les suivants :

- Mars-mai
- Juin-août
- Sept-nov
- Déc-févr

La `base_date` du 15 mai segmente ensuite le trimestre mars-mai en fixant sa limite de fin au 15 mai.

Diagramme de la fonction `inmonthstodate` avec mars défini comme le premier mois de l'année.

Par conséquent, toute transaction qui se produit entre le 1er mars et le 15 mai renverra un résultat booléen TRUE, tandis que les transactions avec des dates en dehors de ces limites renverront une valeur FALSE.

Exemple 4 – Exemple de graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, l'ensemble de données est inchangé et chargé dans l'application. La tâche consiste à créer un calcul qui détermine si des transactions ont eu lieu au cours du même trimestre que celui du 15 mai sous forme de mesure dans un graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
date
```

Pour calculer si des transactions ont eu lieu au cours du même trimestre que celui du 15 mai, créez la mesure suivante :

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

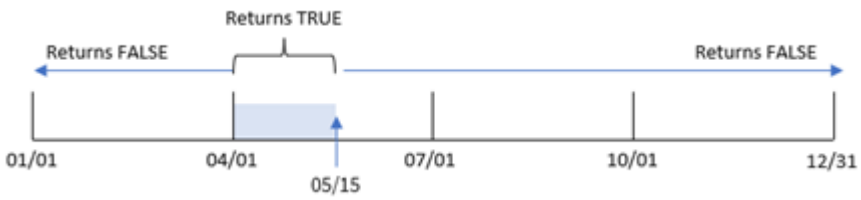
Tableau de résultats

date	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La mesure `'in_months_to_date` est créée dans le graphique à l'aide de la fonction `inmonthstodate()`.

Le premier argument fourni est `3`, divisant l'année en segments trimestriels. Le deuxième argument identifie le champ en cours d'évaluation. Le troisième argument est une date codée en dur du 15 mai, qui est la `base_date` qui définit la limite de fin du segment. Le dernier argument est un argument `period_no` défini sur `0`.

Diagramme de la fonction `inmonthstodate` avec des segments trimestriels.



Toute transaction qui se produit entre le 1er avril et le 15 mai renverra un résultat booléen TRUE. Les dates des transactions en dehors de ce segment renverront FALSE.

Exemple 5 – scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Sales`'. La table contient les champs suivants :

- ID de produit
- Type de produit
- Date de vente
- Prix de vente

L'utilisateur final souhaite un graphique qui affiche, par type de produit, les produits vendus au cours de la période jusqu'au 24 décembre 2022. L'utilisateur souhaite pouvoir définir la longueur de cette période.

Script de chargement

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,sales_date,sales_price
8188,product A,'9/19/2022',37.23
8189,product D,'10/27/2022',17.17
8190,product C,'10/30/2022',88.27
8191,product B,'10/31/2022',57.42
8192,product D,'11/16/2022',53.80
8193,product D,'11/28/2022',82.06
8194,product A,'12/2/2022',40.39
8195,product B,'12/5/2022',87.21
8196,product C,'12/15/2022',95.93
8197,product B,'12/16/2022',45.89
8198,product C,'12/19/2022',36.23
8199,product D,'12/22/2022',25.66
8200,product D,'12/23/2022',82.77
8201,product A,'12/24/2022',69.98
8202,product A,'12/24/2022',76.11
8203,product B,'12/26/2022',25.12
8204,product B,'12/27/2022',46.23
8205,product B,'12/27/2022',84.21
8206,product C,'12/28/2022',96.24
8207,product D,'12/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille.

Au début du script de chargement, une variable, `vPeriod`, est créée ; elle est liée au contrôle d'entrée de variable.

Procédez comme suit :

1. Dans le panneau des ressources, cliquez sur **Objets personnalisés**.
2. Sélectionnez **Qlik Dashboard bundle** et ajoutez une **Entrée de variable** à votre feuille.
3. Saisissez un titre pour le graphique.
4. Sous **Variable**, sélectionnez **vPeriod** comme nom et définissez l'objet de sorte qu'il s'affiche sous forme de **Liste déroulante**.
5. Sous **Valeurs**, cliquez sur les valeurs **Dynamiques**. Saisissez les éléments suivants : `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Ajoutez une nouvelle table à la feuille.
7. Dans le panneau des propriétés, sous **Données**, ajoutez `product_type` comme dimension.
8. Ajoutez l'expression suivante comme mesure : `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Définissez le **Formatage des nombres** de la mesure sur **Money** (Devise).

Tableau de résultats

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))
product A	$186.48
product B	$190.52
product C	$220.43
product D	$261.46

La fonction `inmonthstodate()` utilise l'entrée utilisateur comme argument pour définir la taille du segment de début de l'année.

La fonction transmet la date de vente de chacun des produits comme deuxième argument de la fonction `inmonthstodate()`. Si vous utilisez le 24 décembre comme troisième argument dans la fonction `inmonthstodate()`, les produits dont les dates de vente tombent pendant la période définie jusqu'au 24 décembre inclus renvoient une valeur booléenne TRUE. La fonction sum additionne les ventes de ces produits.

inmonthtodate

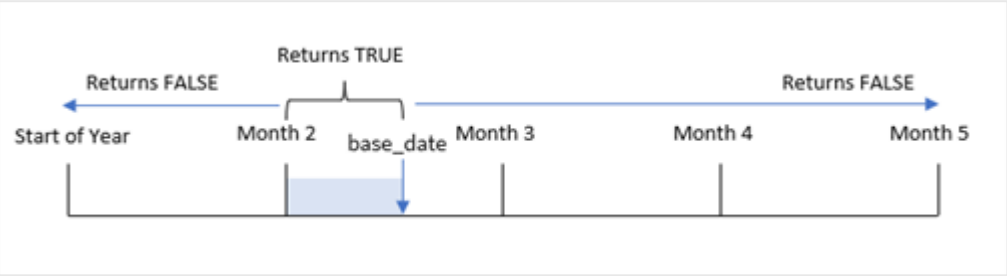
Renvoie la valeur True si l'argument **date** se trouve dans la partie du mois comprenant l'argument **basedate** jusqu'à la dernière milliseconde spécifiée dans **basedate**.

Syntaxe :

```
InMonthToDate (timestamp, base_date, period_no)
```

Type de données renvoyé : booléen

Diagramme de la fonction `inmonthtodate`.



La fonction `inmonthtodate()` identifie un mois sélectionné comme segment. La limite de début est le début du mois. La limite de fin peut être définie sur une date ultérieure au cours du mois. Il est ensuite déterminé si un ensemble de dates tombent dans ce segment ou non, renvoyant une valeur booléenne TRUE ou FALSE.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer le mois.
period_no	Il est possible de décaler le mois à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique le mois comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les mois passés tandis que les valeurs positives désignent les mois à venir.

Cas d'utilisation

La fonction `inmonthtodate()` renvoie un résultat booléen. Ce type de fonction est généralement utilisé comme condition dans une `if expression`. La fonction `inmonthtodate()` renvoie une agrégation ou un calcul qui dépend du fait qu'une date s'est produite ou non au cours du mois jusqu'à la date en question incluse.

Par exemple, la fonction `inmonthtodate()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'un mois jusqu'à une date spécifique.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Renvoie True.
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Renvoie False.
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Renvoie True.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Un champ date est fourni dans la variable système `DateFormat` au format (`MM/DD/YYYY`).
- Instruction LOAD précédente contenant les éléments suivants :
    - La fonction `inmonthtodate()` définie comme le champ '`in_month_to_date`'. Cela détermine les transactions qui ont eu lieu entre le 1er juillet et le 26 juillet 2022.

Script de chargement

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

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', 0) as in_month_to_date
    ;
Load
*
Inline
[
```

```
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_month_to_date`

Tableau de résultats

date	in_month_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1

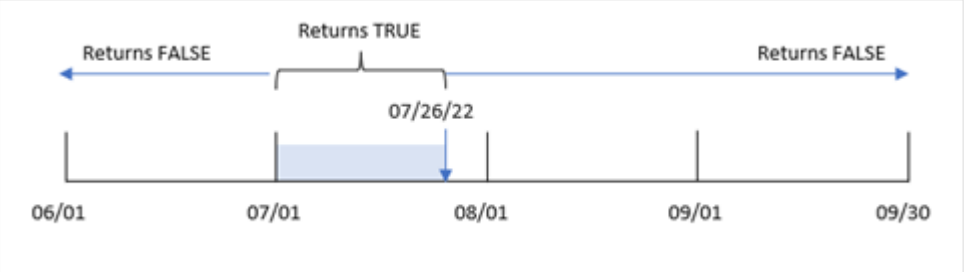
date	in_month_to_date
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ '`in_month_to_date`' est créé dans l'instruction LOAD précédente à l'aide de la fonction `inmonthtodate()`.

Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur, le 26 juillet, qui est la `base_date`. Cet argument `base_date` identifie le mois segmenté et la limite de fin de ce segment.

Un argument `period_no` égal à `0` constitue l'argument final, ce qui signifie que la fonction ne compare pas les mois précédant ou suivant le mois segmenté.

Diagramme de la fonction `inmonthtodate` sans argument supplémentaire.



En conséquence, toute transaction qui se produit entre le 1er juillet et le 26 juillet renvoie un résultat booléen TRUE. Toute transaction qui se produit en juillet après le 26 juillet renvoie un résultat booléen FALSE, comme toute transaction de tout autre mois de l'année.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, la tâche consiste à créer un champ, ‘six_months_prior’, qui détermine les transactions qui ont eu lieu un semestre complet avant le 1er juillet et le 26 juillet.

Script de chargement

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

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', -6) as six_months_prior
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- six_months_prior

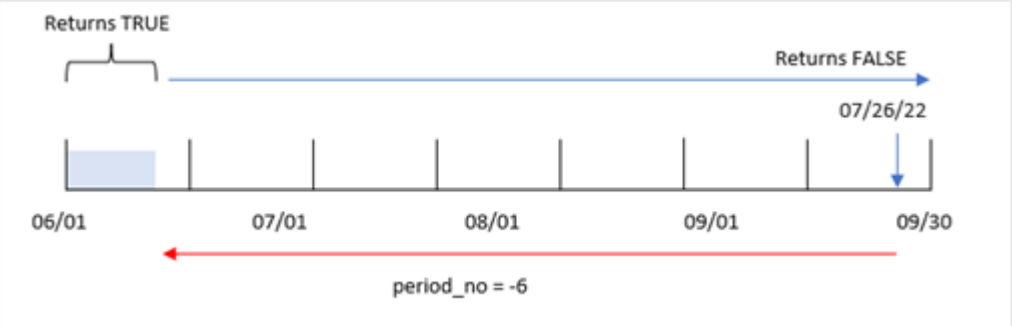
Tableau de résultats

date	six_months_prior
1/7/2022	-1
1/19/2022	-1

date	six_months_prior
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Avec l'utilisation de `-6` comme argument `period_no` dans la fonction `inmonthtodate()`, les limites du segment mensuel de comparaison sont décalées de six mois. Initialement, le segment mensuel va du 1er juillet au 26 juillet. L'argument `period_no` décale alors ce segment de six mois négatifs et les limites de dates sont décalées et tombent entre le 1er janvier et le 26 janvier.

Diagramme de la fonction `inmonthtodate` avec l'argument period_no défini sur -6.



En conséquence, toute transaction qui se produit entre le 1er janvier et le 26 janvier renverra un résultat booléen TRUE.

Exemple 3 - exemple graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, l'ensemble de données est inchangé et chargé dans l'application. La tâche consiste à créer un calcul qui détermine si des transactions ont eu lieu entre le 1er juillet et le 26 juillet sous forme de mesure dans un graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
date
```

Pour calculer si des transactions ont eu lieu entre le 1er juillet et le 26 juillet, créez la mesure suivante :

```
=inmonthtodate(date,'07/26/2022', 0)
```

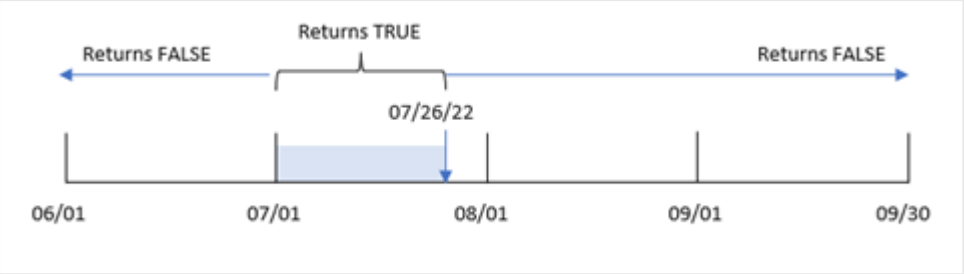
Tableau de résultats

date	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La mesure du champ '`in_month_to_date`' est créée dans le graphique à l'aide de la fonction `inmonthtodate ()`.

Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur, le 26 juillet, qui est la `base_date`. Cet argument base_date identifie le mois segmenté et la limite de fin de ce segment. Le dernier argument est un argument `period_no` défini sur 0. Cela signifie que la fonction ne compare pas les mois précédant ou suivant le mois segmenté.

Diagramme de la fonction `inmonthtodate` sans argument supplémentaire.



En conséquence, toute transaction qui se produit entre le 1er juillet et le 26 juillet renvoie un résultat booléen TRUE. Toute transaction qui se produit en juillet après le 26 juillet renvoie un résultat booléen FALSE, comme toute transaction de tout autre mois de l'année.

Exemple 4 – scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Products`'. La table contient les champs suivants :

- ID de produit
- Manufacture date
- Cost price

En raison d'une erreur d'équipement, les produits fabriqués en juillet 2022 étaient défectueux. Le problème a été résolu le 27 juillet 2022.

L'utilisateur final souhaite un graphique qui affiche, par mois, l'état des produits fabriqués 'defective' (défectueux) (valeur booléenne TRUE) ou 'faultless' (corrects) (valeur booléenne FALSE) et le coût des produits fabriqués au cours de ce mois-là.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
```

```
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Pour calculer le coût total (sum) des produits, créez la mesure suivante :

```
=sum(cost_price)
```

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69
Mar 2022	Faultless	$53.80
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$144.66
Jul 2022	Faultless	$69.98
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

La fonction `inmonthtodate()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits.

Pour les dates qui renvoient une valeur booléenne TRUE, le produit est identifié comme ‘Defective’ (défectueux). Tout produit renvoyant une valeur FALSE, et par conséquent non fabriqué au cours du mois jusqu'au 26 juillet inclus, est identifié comme ‘Faultless’ (correct).

inquarter

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans le trimestre comprenant l'argument **base_date**.

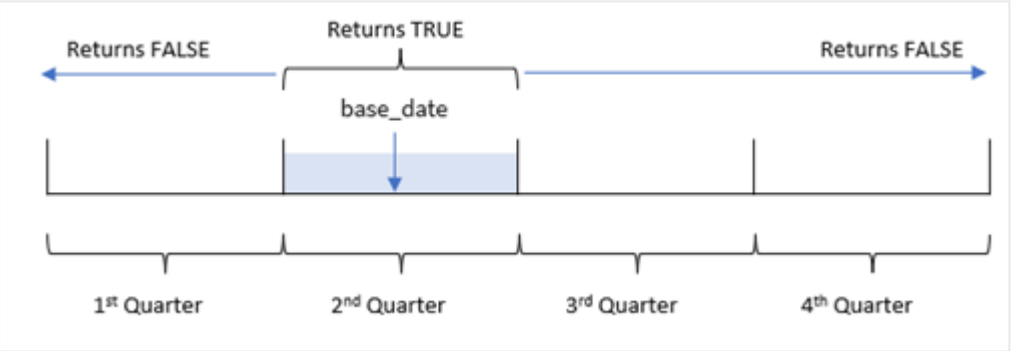
Syntaxe :

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Diagramme de la plage de la fonction `inquarter()`



En d'autres termes, la fonction `inquarter()` divise l'année en quatre trimestres égaux entre le 1er janvier et le 31 décembre. Vous pouvez utiliser l'argument `first_month_of_year` pour modifier le mois considéré comme le premier dans votre application, et les trimestres changeront en fonction de cet argument. La `base_date` ; la fonction identifie le trimestre à utiliser comme comparateur pour la fonction. Pour finir, la fonction renvoie un résultat booléen lors de la comparaison des valeurs de date à ce segment trimestriel.

Cas d'utilisation

La fonction `inquarter()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. Cela renvoie une agrégation ou un calcul qui dépend du fait qu'une date s'est produite ou non au cours du trimestre sélectionné.

Par exemple, la fonction `inquarter()` peut permettre d'identifier l'ensemble des équipements fabriqués au cours d'un segment trimestriel en fonction des dates de fabrication des équipements.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.

Argument	Description
base_date	Date utilisée pour évaluer le trimestre.
period_no	Il est possible de décaler le trimestre à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique le trimestre comprenant l'argument **base_date**. Les valeurs négatives de l'argument **period_no** indiquent les trimestres passés tandis que les valeurs positives désignent les trimestres à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres

régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
inquarter ('01/25/2013', '01/01/2013', 0)	Renvoie TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Renvoie FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Renvoie FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Renvoie TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Renvoie FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Renvoie TRUE

Exemple 1 - aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inquarter()` définie comme le champ '`in_quarter`' et qui détermine les transactions qui ont eu lieu au cours du même trimestre que celui du 15 mai 2022.

Script de chargement

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

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
```

```
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_quarter`

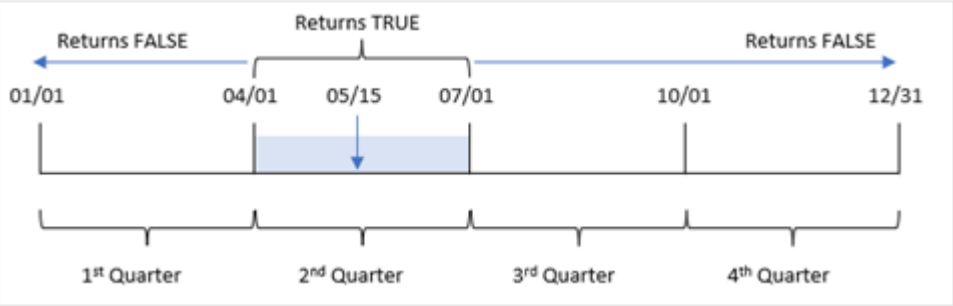
Tableau de résultats

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

date	in_quarter
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ '`in_quarter`' est créé dans l'instruction LOAD précédente à l'aide de la fonction `inquarter()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 15 mai qui identifie le trimestre à définir comme comparateur. Un argument `period_no` égal à `0` constitue l'argument final et garantit que la fonction `inquarter()` ne compare pas les trimestres précédant ou suivant le trimestre segmenté.

Diagramme de la fonction `inquarter()` avec le 15 mai comme date de référence



Toute transaction qui se produit entre le 1er avril et la fin du 30 juin renvoie un résultat booléen TRUE.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inquarter()` définie comme le champ '`previous_quarter`', et qui détermine si les transactions ont eu lieu au cours du trimestre précédant le trimestre du 15 mai 2022.

Script de chargement

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

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', -1) as previous_qtr
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_qtr`

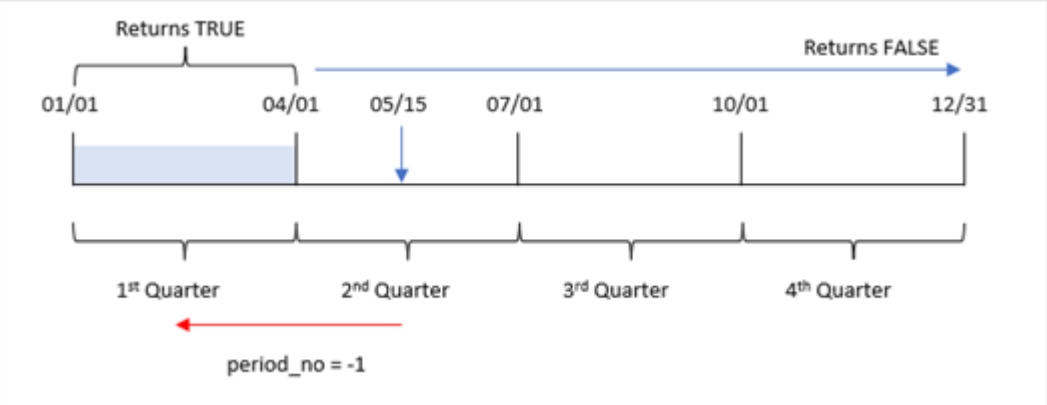
Tableau de résultats

date	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1

date	previous_qtr
3/16/2022	-1
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Avec l'utilisation de `-1` comme argument `period_no` dans la fonction `inquarter()`, les limites du trimestre de comparaison sont décalées d'un trimestre complet en arrière. Le 15 mai tombe pendant le deuxième trimestre de l'année, et, par conséquent, le segment correspond initialement au trimestre du 1er avril au 30 juin. L'argument `period_no` décale ce segment de trois mois négatifs, faisant passer les limites de date au trimestre du 1er janvier au 30 mars.

Diagramme de la fonction `inquarter()` avec le 15 mai comme date de référence



Par conséquent, toute transaction effectuée entre le 1er janvier et le 30 mars renverra un résultat booléen TRUE.

Exemple 3 - first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inquarter()` définie comme le champ '`in_quarter`' et qui détermine les transactions qui ont eu lieu au cours du même trimestre que celui du 15 mai 2022.

Cependant, dans cet exemple, la stratégie organisationnelle exige que mars soit le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0, 3) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
```

```
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

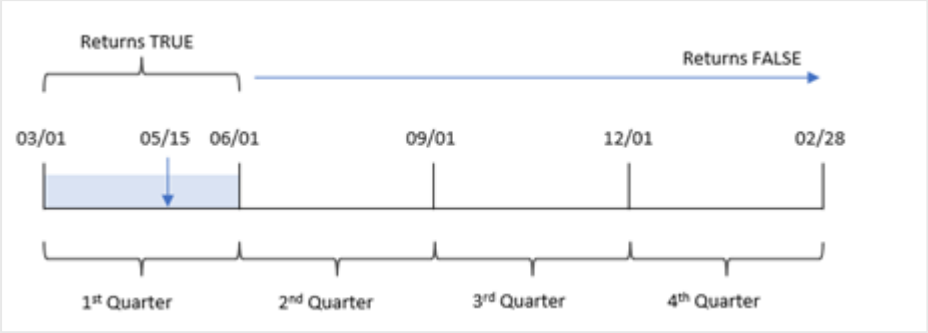
- `date`
- `previous_qtr`

Tableau de résultats

date	previous_qtr
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Si on utilise `3` comme argument `first_month_of_year` dans la fonction `inquarter()`, l'année commence le 1er mars, puis elle est divisée en trimestres. Par conséquent, les segments trimestriels sont mars-mai, juin-août, sept-nov et déc-févr. La `base_date` du 15 mai défini le trimestre mars-mai comme le trimestre de comparaison de la fonction.

Diagramme de la fonction `inquarter()` avec mars défini comme le premier mois de l'année



Par conséquent, toute transaction effectuée entre le 1er mars et le 31 mai renverra un résultat booléen TRUE.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inquarter()` définie comme le champ '`in_quarter`' et qui détermine les transactions qui ont eu lieu au cours du même trimestre que celui du 15 mai 2022.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
```

```
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `date`

Créez la mesure suivante pour calculer si des transactions ont eu lieu au cours du même trimestre que celui du 15 mai :

```
=inquarter(date,'05/15/2022', 0)
```

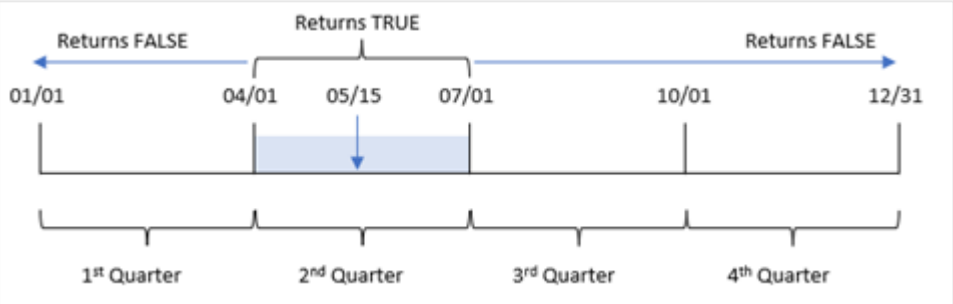
Tableau de résultats

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0

date	in_quarter
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La mesure '`in_quarter`' est créée dans le graphique à l'aide de la fonction `inquarter()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 15 mai qui identifie le trimestre à définir comme comparateur. Un argument `period_no` égal à `0` constitue l'argument final et garantit que la fonction `inquarter()` ne compare pas les trimestres précédant ou suivant le trimestre segmenté.

Diagramme de la fonction `inquarter()` avec le 15 mai comme date de référence



Toute transaction qui se produit entre le 1er avril et la fin du 30 juin renvoie un résultat booléen TRUE.

Exemple 5 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Products`'.
- La table contient les champs suivants :
  - product ID
  - product type
  - manufacture date
  - cost price

Il a été identifié que, suite à une erreur d'équipement, des produits fabriqués au cours du trimestre du 15 mai 2022 étaient défectueux. L'utilisateur final souhaite un graphique qui affiche, par nom de trimestre, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués ce trimestre-là.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
=quartername(manufacture_date)
```

Créez les mesures suivantes :

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, pour identifier les produits défectueux et les produits corrects via la fonction `inquarter()`.
- `=sum(cost_price)`, pour afficher la somme du coût de chaque produit.

Procédez comme suit :

1. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).
2. Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

quartername (manufacture_date)	=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')	Sum(cost_ price)
Jan-Mar 2022	Faultless	253.89
Apr-Jun 2022	Defective	351.48
Jul-Sep 2022	Faultless	446.31
Oct-Dec 2022	Faultless	163.91

La fonction `inquarter()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué au cours du trimestre qui contient le 15 mai, la fonction `inquarter()` renvoie une valeur booléenne TRUE et identifie les produits comme 'Defective' (défectueux). Tout produit renvoyant une valeur FALSE, et, par conséquent, non fabriqué au cours de ce trimestre-là, est identifié comme 'Faultless' (correct).

inquartertodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie du trimestre comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

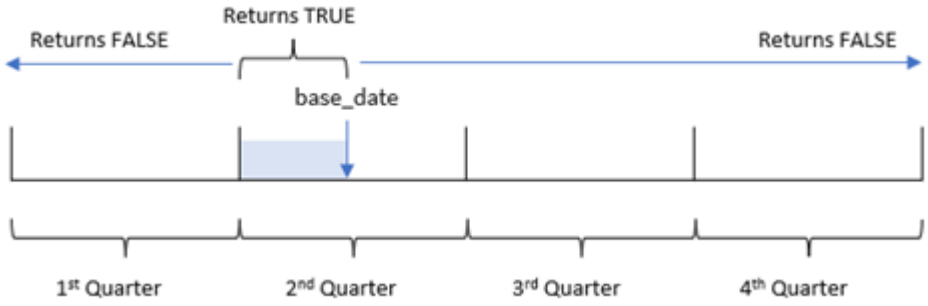
Syntaxe :

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Type de données renvoyé : booléen

> *Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.*

Diagramme de la fonction inquartertodate



La fonction `inquartertodate()` divise l'année en quatre trimestres égaux entre le 1er janvier et le 31 décembre (ou le début de l'année défini par l'utilisateur et sa date de fin correspondante). Avec `base_date`, la fonction segmentera ensuite un trimestre donné, la valeur `base_date` identifiant à la fois le trimestre et la date maximale autorisée pour ce segment de trimestre. Pour finir, la fonction renvoie un résultat booléen lors de la comparaison des valeurs de date prescrites à ce segment.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer le trimestre.
period_no	Il est possible de décaler le trimestre à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique le trimestre comprenant l'argument **base_date**. Les valeurs négatives de l'argument **period_no** indiquent les trimestres passés tandis que les valeurs positives désignent les trimestres à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `inquartertodate()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if`. La fonction `inquartertodate()` serait utilisée pour renvoyer une agrégation ou un calcul suivant qu'une date évaluée s'est produite ou non au cours du trimestre jusqu'à la date en question incluse.

Par exemple, la fonction `inquartertodate()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'un trimestre jusqu'à une date spécifique.

Exemples de fonction

Exemple	Résultat
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Renvoie`TRUE`, car la valeur de `timestamp`, 01/25/2013, se situe dans la période de trois mois de 01/01/2013 à 03/25/2013 dans laquelle figure la valeur de `base_date`, 03/25/2013.
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Renvoie `FALSE`, car 04/26/2013 se trouve hors de la même période que celle indiquée dans l'exemple précédent.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Renvoie `TRUE`, car la valeur de `period_no`, -1, décale la période de recherche d'une période de trois mois (d'un trimestre de l'année) en arrière. La période de recherche devient ainsi du 01/01/2013 au 03/09/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Renvoie `TRUE`, car la valeur de `first_month_of_year`est définie sur 2, ce qui fait que la période de recherche devient du 02/01/2006 au 04/15/2006 au lieu du 04/01/2006 au 04/15/2006.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `in_quarter_to_date`, qui détermine les transactions qui ont eu lieu au cours du trimestre jusqu'au 15 mai 2022.

Script de chargement

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

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

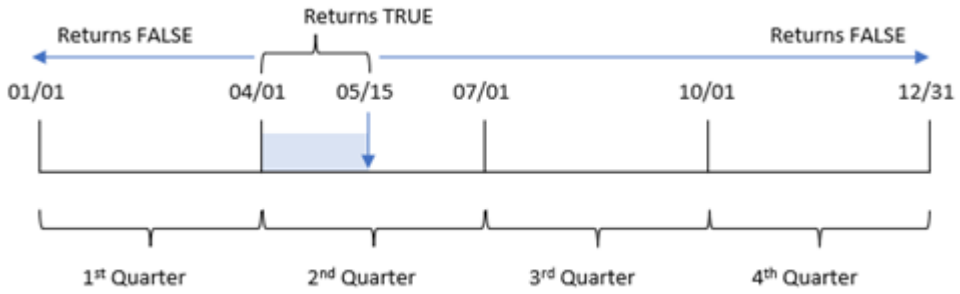
- `date`
- `in_quarter_to_date`

Tableau de résultats

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Le champ `in_quarter_to_date` est créé dans l'instruction LOAD précédente à l'aide de la fonction `inquartertodate()`. Le premier argument fourni identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 15 mai, qui est la valeur `base_date` qui identifie le trimestre à segmenter et définit la limite de fin de ce segment. Une valeur `period_no` de 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les trimestres précédant ou suivant le trimestre segmenté.

Diagramme de la fonction inquartertodate, sans argument supplémentaire



Toute transaction qui se produit entre le 1er avril et le 15 mai renvoie un résultat booléen `TRUE`. Les dates de transaction du 16 mai et ultérieures renverront `FALSE`, comme toutes les transactions avant le 1er avril.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_qtr_to_date`, qui détermine les transactions qui ont eu lieu un trimestre complet avant la fin du segment de trimestre le 15 mai 2022.

Script de chargement

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

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', -1) as previous_qtr_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
```

```
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_qtr_to_date`

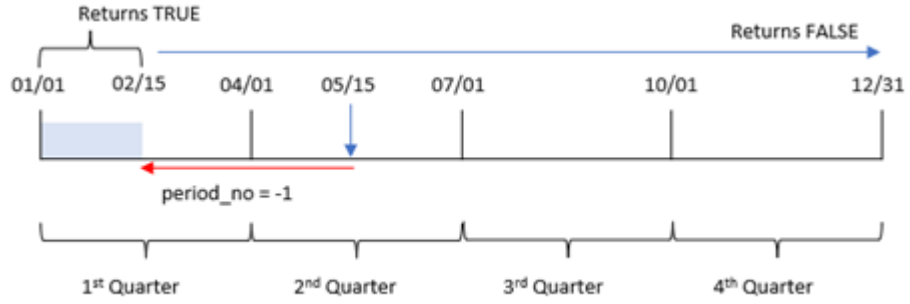
Tableau de résultats

date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0

date	previous_qtr_to_date
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Une valeur `period_no` égale à -1 indique que la fonction `inquartertodate ()` compare le segment de trimestre d'entrée au trimestre précédent. Le 15 mai tombe dans le deuxième trimestre de l'année, si bien que le segment correspond initialement à la période entre le 1er avril et le 15 mai. La valeur `period_no` décale ensuite ce segment de trois mois plus tôt, faisant passer les limites de date à la période du 1er janvier au 15 février.

Diagramme de la fonction inquartertodate, exemple period_no



Par conséquent, toute transaction effectuée entre le 1er janvier et le 15 février renverra un résultat booléen TRUE.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `in_quarter_to_date`, qui détermine les transactions qui ont eu lieu au cours du même trimestre jusqu'au 15 mai 2022.

Dans cet exemple, nous définissons mars comme le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0,3) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_quarter_to_date`

Tableau de résultats

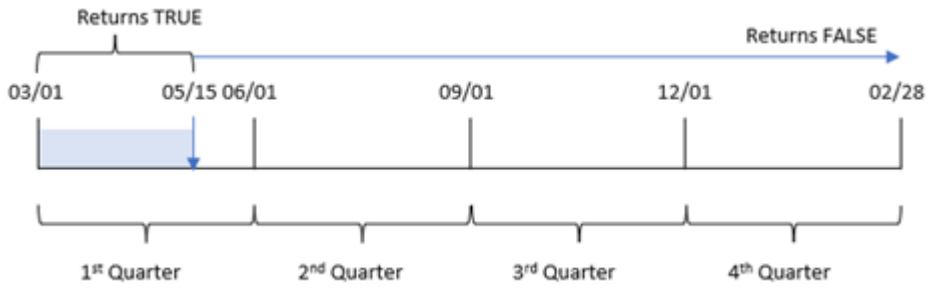
date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Si on utilise 3 comme argument `first_month_of_year` dans la fonction `inquartertodate()`, la fonction commence l'année le 1er mars, puis divise l'année en trimestres. Par conséquent, les segments de trimestre sont les suivants :

- mars à mai
- juin à août
- septembre à novembre
- décembre à février

La valeur `base_date` du 15 mai segmente ensuite le trimestre de mars à mai en fixant sa limite de fin au 15 mai.

Diagramme de la fonction inquartertodate, exemple first_month_of_year



Par conséquent, toute transaction qui se produit entre le 1er mars et le 15 mai renverra un résultat booléen TRUE, tandis que les transactions avec des dates en dehors de ces limites renverront une valeur FALSE.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu le même trimestre que celui du 15 mai est créé sous forme de mesure dans l'objet graphique.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.

Créez la mesure suivante :

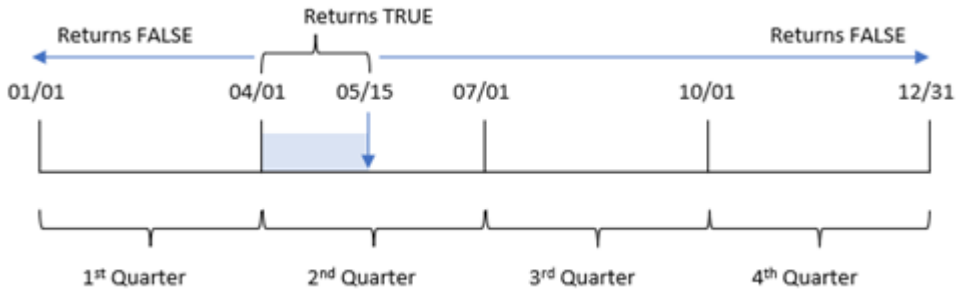
```
=inquartertodate(date,'05/15/2022', 0)
```

Tableau des résultats

date	=inquartertodate(date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La mesure `in_quarter_to_date` est créée dans un objet graphique à l'aide de la fonction `inquartertodate()`. Le premier argument est le champ de date en cours d'évaluation. Le deuxième argument est une date codée en dur du 15 mai, qui est la valeur `base_date` qui identifie le trimestre à segmenter et définit la limite de fin de ce segment. Une valeur `period_no` de 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les trimestres précédant ou suivant le trimestre segmenté.

Diagramme de la fonction inquartertodate, exemple objet graphique



Toute transaction qui se produit entre le 1er avril et le 15 mai renvoie un résultat booléen `TRUE`. Les transactions du 16 mai et ultérieures renverront `FALSE`, comme toutes les transactions avant le 1er avril.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Products`.
- Informations concernant l'ID du produit, la date de fabrication et le prix de revient.

Le 15 mai 2022, une erreur d'équipement a été identifiée dans le processus de fabrication et résolue. Les produits fabriqués au cours du trimestre en question jusqu'à cette date seront défectueux. L'utilisateur final souhaite un objet graphique qui affiche, par nom de trimestre, l'état du produit ('defective' (défectueux) ou 'faultless' (correct)) et le coût des produits fabriqués au cours du trimestre en question jusqu'à aujourd'hui.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
```

```
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau. Créez une dimension pour afficher les noms de trimestre :
   `=quartername(manufacture_date)`
2. Ensuite, créez une dimension pour identifier les produits défectueux et les produits corrects :
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Créez une mesure pour additionner les valeurs `cost_price` des produits :
   `=sum(cost_price)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

quartername (manufacture_date)	if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')	Sum(cost_ price)
Jan-Mar 2022	Faultless	$253.89
Apr-Jun 2022	Faultless	$229.03
Apr-Jun 2022	Defective	$122.45
Jul-Sep 2022	Faultless	$446.31
Oct-Dec 2022	Faultless	$163.91

La fonction `inquartertodate()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Les produits qui renvoient une valeur booléenne `TRUE` sont identifiés comme `'Defective'`. Tout produit renvoyant une valeur `FALSE` et par conséquent non fabriqué au cours du trimestre jusqu'au 15 mai inclus est identifié comme `'Faultless'`.

inweek

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la semaine comprenant l'argument **base_date**.

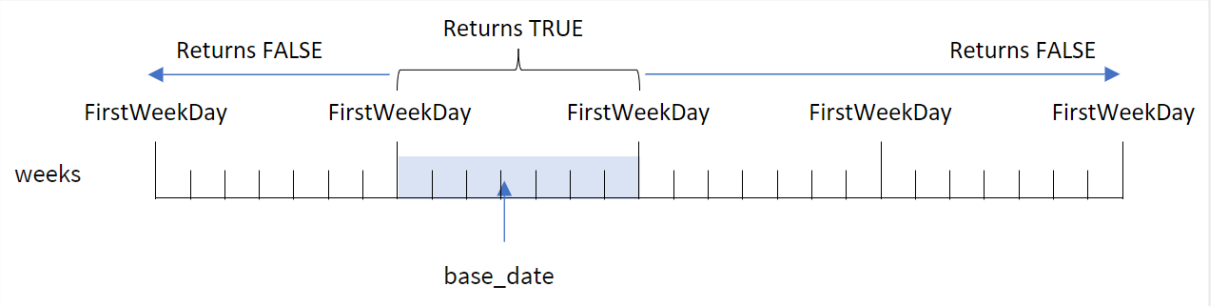
Syntaxe :

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Diagramme de la plage de la fonction `inweek()`



La fonction `inweek()` utilise l'argument `base_date` pour identifier la période de sept jours de la date. Le jour de début de la semaine est basé sur la variable système `FirstWeekDay`. Cependant, vous pouvez également modifier le premier jour de la semaine en utilisant l'argument `first_week_day` de la fonction `inweek()`.

Une fois la semaine sélectionnée définie, la fonction renverra des résultats booléens lors de la comparaison des valeurs de date prescrites à ce segment hebdomadaire.

Cas d'utilisation

La fonction `inweek()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. La fonction `inweek()` renvoie une agrégation ou un calcul qui dépend du fait qu'une date évaluée est tombée ou non pendant la semaine contenant la date sélectionnée de l'argument `base_date`.

Par exemple, la fonction `inweek()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'une semaine spécifique.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer la semaine.

Argument	Description
period_no	Il est possible de décaler la semaine à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique la semaine comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines passées tandis que les valeurs positives désignent les semaines à venir.
first_week_day	Par défaut, le premier jour de la semaine est le dimanche (tel que déterminé par la variable système FirstWeekDay), commençant à minuit entre le samedi et le dimanche. Le paramètre **first_week_day** remplace la variable **FirstWeekDay**. Pour que la semaine commence un autre jour, spécifiez un indicateur compris entre 0 et 6.

first_week_day values

Jour	Valeur
Lundi	0
Mardi	1
Mercredi	2
Jeudi	3
Vendredi	4
Samedi	5
Dimanche	6

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inweek ('01/12/2006', '01/14/2006', 0)`	Renvoie TRUE

Exemple	Résultat
`inweek ('01/12/2006', '01/20/2006', 0 )`	Renvoie FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Renvoie FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	Renvoie TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Renvoie FALSE, car la valeur `first_week_day` est spécifiée sur 3 (jeudi), ce qui fait de la date du 01/12/2006 le premier jour de la semaine suivant la semaine contenant le 01/09/2006.

Ces rubriques peuvent vous aider à utiliser cette fonction :

Rubriques connexes

Rubrique	Indicateur/valeur par défaut	Description
FirstWeekDay (page 221)	6 / Dimanche	Définit le jour de début de chaque semaine.

Exemple 1 - aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour le mois de janvier 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur `6` (dimanche).
- Chargement précédent contenant les éléments suivants :
  - Fonction `inweek()`, définie comme le champ '`in_week`', qui détermine les transactions qui ont eu lieu au cours de la semaine du 14 janvier 2022.
  - Fonction `weekday()`, définie comme le champ '`week_day`', qui indique le jour de la semaine correspondant à chaque date.

Script de chargement

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `week_day`
- `in_week`

Tableau de résultats

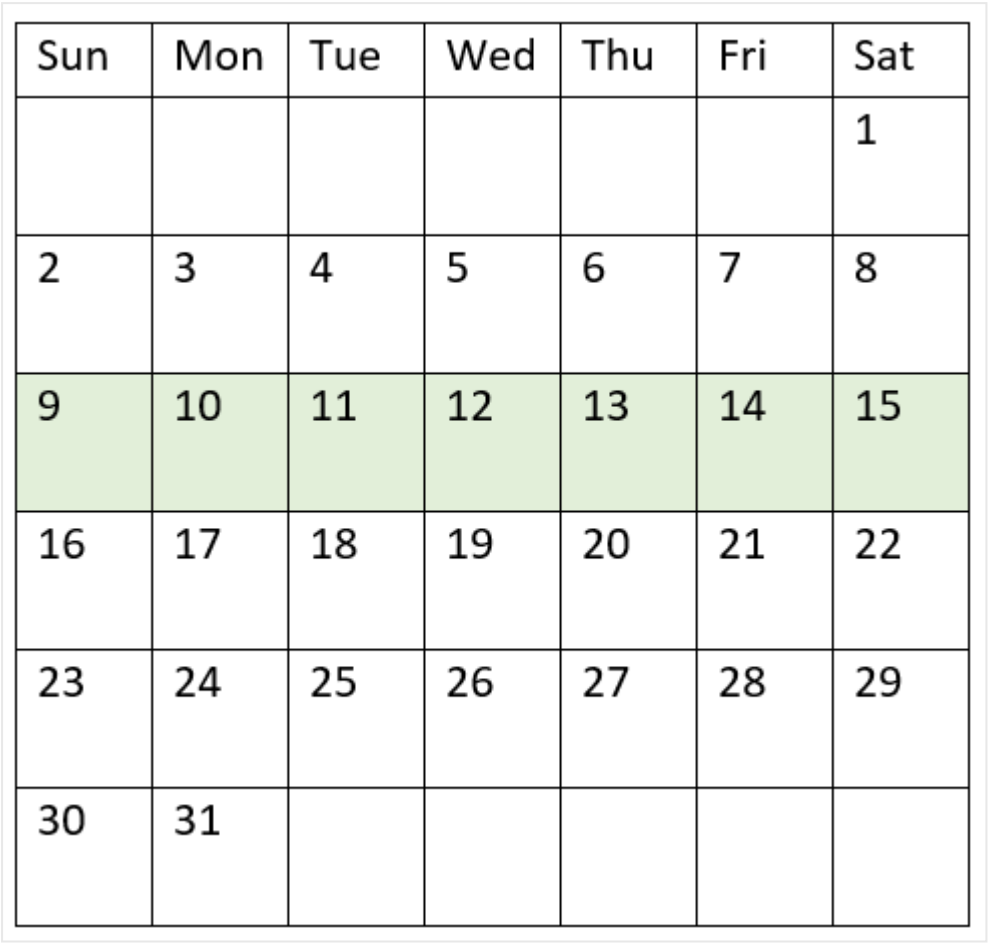
date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0

date	week_day	in_week
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Le champ 'in_week' est créé dans l'instruction LOAD précédente à l'aide de la fonction `inweek()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 14 janvier, qui est la `base_date`. L'argument `base_date` fonctionne avec la variable système `FirstWeekDay` pour identifier la semaine de comparaison. Un argument `period_no` égal à `0` – qui signifie que la fonction ne compare pas les semaines précédant ou suivant la semaine segmentée – constitue l'argument final.

La variable système `FirstWeekDay` détermine que les semaines commencent un dimanche et se terminent un samedi. Par conséquent, janvier serait divisé en semaines selon le diagramme ci-dessous, les dates comprises entre le 9 et le 15 janvier fournissant la période valide pour le calcul `inweek()` :

Diagramme de calendrier avec la plage de la fonction `inweek()` en surbrillance



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Toute transaction qui se produit entre le 9 janvier et le 15 janvier renvoie un résultat booléen `TRUE`.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur `6` (dimanche).
- Chargement précédent contenant les éléments suivants :
    - Fonction `inweek ()`, définie comme le champ '`prev_week`', qui détermine les transactions qui ont eu lieu une semaine complète avant la semaine du 14 janvier 2022.

- Fonction weekday(), définie comme le champ 'week_day', qui indique le jour de la semaine correspondant à chaque date.

Script de chargement

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', -1) as prev_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

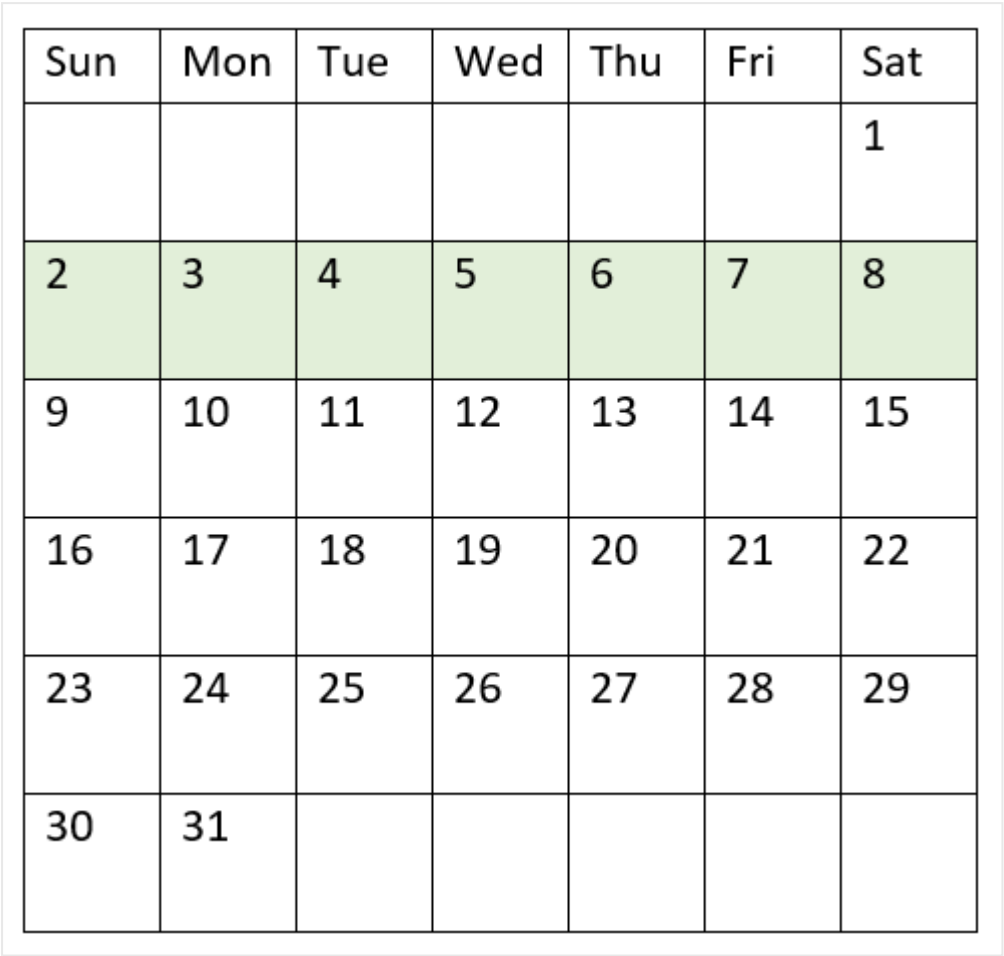
- date
- week_day
- prev_week

Tableau de résultats

date	week_day	prev_week
01/02/2022	Sun	-1
01/05/2022	Wed	-1
01/06/2022	Thu	-1
01/08/2022	Sat	-1
01/09/2022	Sun	0
01/10/2022	Mon	0
01/11/2022	Tue	0
01/12/2022	Wed	0
01/13/2022	Thu	0
01/14/2022	Fri	0
01/15/2022	Sat	0
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Si on applique la valeur `-1` à l'argument `period_no` dans la fonction `inweek()`, les limites de la semaine de comparaison sont décalées de sept jours complets en arrière. Avec un argument `period_no` égal à `0`, la semaine serait comprise entre le 9 et le 15 janvier. Mais, dans cet exemple, l'argument `period_no` défini sur `-1` décale les limites de début et de fin de ce segment d'une semaine en arrière. Les limites des dates deviennent du 2 janvier au 8 janvier.

Diagramme de calendrier avec la plage de la fonction `inweek()` en surbrillance



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Par conséquent, toute transaction effectuée entre le 2 janvier et le 8 janvier renverra un résultat booléen TRUE.

Exemple 3 - first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur `6` (dimanche).
- Chargement précédent contenant les éléments suivants :

- Fonction `inweek()`, définie comme le champ '`in_week`', qui détermine les transactions qui ont eu lieu au cours de la semaine du 14 janvier 2022.
- Fonction `weekday()`, définie comme le champ '`week_day`', qui indique le jour de la semaine correspondant à chaque date.

Script de chargement

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0, 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `week_day`
- `in_week`

Tableau de résultats

date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	0
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	-1
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Si on applique `0` à l'argument `first_week_day` dans la fonction `inweek()`, l'argument remplace la variable système `FirstWeekDay` et définit le lundi comme le premier jour de la semaine.

Diagramme de calendrier avec la plage de la fonction `inweek()` en surbrillance

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Par conséquent, toute transaction effectuée entre le 10 et le 16 janvier renverra un résultat booléen TRUE.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Créez une mesure dans le tableau de résultats pour déterminer les transactions qui ont eu lieu au cours de la semaine du 14 janvier 2022.

Script de chargement

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
```

```
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `date`

Créez les mesures suivantes :

- `=inweek (date,'01/14/2022',0)`, pour calculer si des transactions ont eu lieu au cours de la même semaine que celle du 14 janvier.
- `=weekday(date)`, pour indiquer quel jour de la semaine correspond à chaque date.

Tableau de résultats

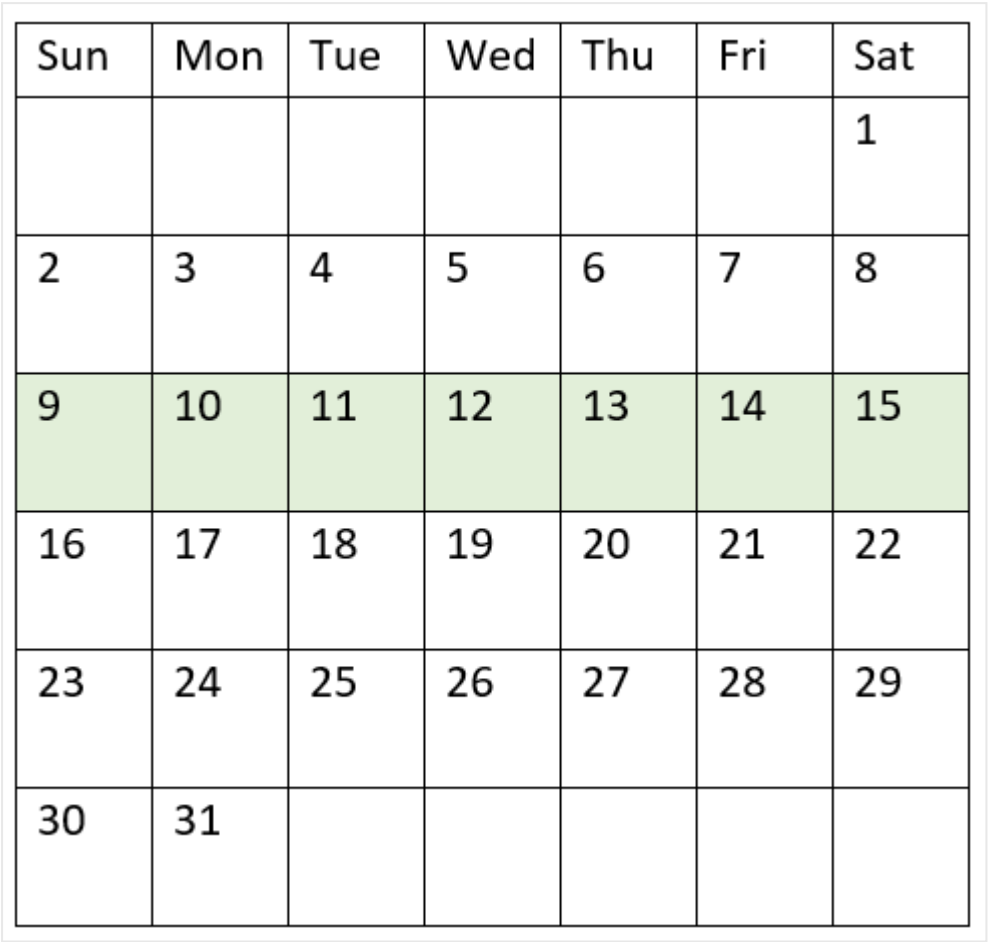
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1
01/11/2022	Tue	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

La mesure ‘`in_week`’ est créée dans le graphique à l'aide de la fonction `inweek()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 14 janvier, qui est la `base_date`. L'argument `base_date` fonctionne avec la variable système `FirstWeekDay` pour identifier la semaine de comparaison. Le dernier argument est un argument `period_no` défini sur `0`.

La variable système `FirstWeekDay` détermine que les semaines commencent un dimanche et se terminent un samedi. Par conséquent, janvier serait divisé en semaines selon le diagramme ci-dessous, les dates comprises entre le 9 et le 15 janvier fournissant la période valide pour le calcul `inweek()` :

Diagramme de calendrier avec la plage de la fonction `inweek()` en surbrillance



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Toute transaction qui se produit entre le 9 janvier et le 15 janvier renvoie un résultat booléen `TRUE`.

Exemple 5 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Products`'.
- La table contient les champs suivants :
  - product ID
  - product type
  - manufacture date
  - cost price

Il a été identifié que suite, à une erreur d'équipement, des produits fabriqués la semaine du 12 janvier étaient défectueux. L'utilisateur final souhaite un graphique qui affiche, par semaine, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués cette semaine-là.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `=weekname(manufacture_date)`

Créez les mesures suivantes :

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, pour identifier les produits défectueux et les produits corrects via la fonction `inweek()`.
- `=sum(cost_price)`, pour afficher la somme du coût de chaque produit.

Procédez comme suit :

1. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).
2. Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

weekname (manufacture_date)	=if(only(inweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	Sum(cost_ price)
2022/02	Faultless	200.09
2022/03	Defective	441.51
2022/04	Faultless	178.41
2022/05	Faultless	231.67
2022/06	Faultless	163.91

La fonction `inweek()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué au cours de la semaine du 12 janvier, la fonction `inweek()` renvoie une valeur booléenne TRUE et identifie les produits comme 'Defective' (défectueux). Tout produit renvoyant une valeur FALSE, et par conséquent non fabriqué au cours de cette semaine-là, est identifié comme 'Faultless' (correct).

inweektodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie de la semaine comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

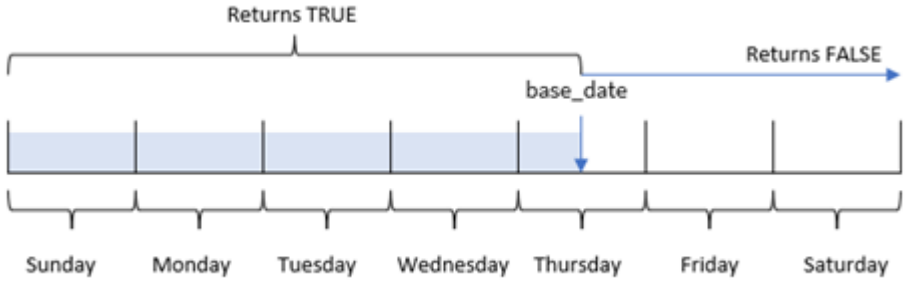
Syntaxe :

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Type de données renvoyé : booléen

> *Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.*

Diagramme de la fonction inweektodate



La fonction `inweektodate()` utilise le paramètre `base_date` pour identifier une date limite maximale d'un segment de semaine, ainsi que sa date correspondante pour le début de la semaine, qui est basée sur la

variable système `FirstWeekDay` (ou le paramètre `first_week_day` défini par l'utilisateur). Une fois ce segment de semaine défini, la fonction renvoie des résultats booléens lors de la comparaison des valeurs de date prescrites à ce segment.

Cas d'utilisation

La fonction `inweektodate()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if`. Cela renvoie une agrégation ou un calcul suivant qu'une date évaluée s'est produite ou non au cours de la semaine en question jusqu'à une date donnée incluse.

Par exemple, la fonction `inweektodate()` peut être utilisée pour calculer toutes les ventes réalisées au cours d'une semaine spécifiée jusqu'à une date spécifique.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer la semaine.
period_no	Il est possible de décaler la semaine à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique la semaine comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines passées tandis que les valeurs positives désignent les semaines à venir.
first_week_day	Par défaut, le premier jour de la semaine est le dimanche (tel que déterminé par la variable système FirstWeekDay), commençant à minuit entre le samedi et le dimanche. Le paramètre **first_week_day** remplace la variable **FirstWeekDay**. Pour que la semaine commence un autre jour, spécifiez un indicateur compris entre 0 et 6. Pour une semaine commençant le lundi et se terminant le dimanche, utilisez un indicateur 0 pour lundi, 1 pour mardi, 2 pour mercredi, 3 pour jeudi, 4 pour vendredi, 5 pour samedi et 6 pour dimanche.

Exemples de fonction

Exemple	Interaction
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Renvoie `TRUE`.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Renvoie `FALSE`.
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Renvoie `FALSE`. Comme `period_no` est défini sur -1, la date réelle par rapport à laquelle l'argument d'horodatage `timestamp` est mesuré est le 01/11/2006.

Exemple	Interaction
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Renvoie `FALSE`, car `first_week_day` est défini sur `3` (un jeudi), ce qui fait du 01/12/2006 le premier jour de la semaine suivant la semaine contenant la date du 01/12/2006.

Ces rubriques peuvent vous aider à utiliser cette fonction :

Rubriques connexes

Rubrique	Indicateur/valeur par défaut	Description
FirstWeekDay (page 221)	6 / Dimanche	Définit le jour de début de chaque semaine.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour le mois de janvier 2022, chargé dans une table appelée `Transactions`.
- Champ de données fourni au format `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`.
- Création d'un champ, `in_week_to_date`, qui détermine les transactions qui ont eu lieu au cours de la semaine jusqu'au 14 janvier 2022.

- Création d'un champ supplémentaire appelé `weekday` via la fonction `weekday()`. Ce nouveau champ est créé pour indiquer quel jour de la semaine correspond à chaque date.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
SET FirstWeekDay=6;
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `week_day`
- `in_week_to_date`

Tableau de résultats

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Le champ `in_week_to_date` est créé dans l'instruction LOAD précédente à l'aide de la fonction `inweektodate()`. Le premier argument fourni identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 14 janvier, qui est la valeur `base_date` qui identifie la semaine à segmenter et définit la limite de fin du segment en question. Une valeur `period_no` égale à 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les semaines précédant ou suivant la semaine segmentée.

La variable système `FirstWeekDay` détermine que les semaines commencent un dimanche et se terminent un samedi. Par conséquent, janvier serait divisé en semaines selon le diagramme ci-dessous, les dates comprises entre le 9 et le 14 janvier fournissant la période valide pour le calcul `inweekdodate()` :

Diagramme de calendrier indiquant les dates de transaction qui renverraient un résultat booléen TRUE

Sun	Mon	Tue	Wed	Thur	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Toute transaction qui se produit entre le 9 janvier et le 14 janvier renvoie un résultat booléen TRUE. Les transactions avant et après les dates renvoient un résultat booléen FALSE.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `prev_week_to_date`, qui détermine les transactions qui ont eu lieu une semaine complète avant la fin du segment de semaine le 14 janvier 2022.
- Création d'un champ supplémentaire appelé `weekday` via la fonction `weekday()`. Cela a pour objectif d'indiquer quel jour de la semaine correspond à chaque date.

Script de chargement

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', -1) as prev_week_to_date
        ;
Load
*
Inline
[
```

```
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `week_day`
- `prev_week_to_date`

Tableau de résultats

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Sun	-1
2022-01-05 01:02:30	Wed	-1
2022-01-06 15:36:20	Thu	-1
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	0
2022-01-11 00:57:13	Tue	0
2022-01-12 09:26:02	Wed	0
2022-01-13 15:05:09	Thu	0
2022-01-14 18:44:57	Fri	0
2022-01-15 06:10:46	Sat	0

date	week_day	prev_week_to_date
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Une valeur `period_no` égale à -1 indique que la fonction `inweektodate ()` compare le segment de trimestre d'entrée à la semaine précédente. Le segment de semaine équivaut initialement aux dates entre le 9 janvier et le 14 janvier. La valeur `period_no` décale ensuite les deux limites de début et de fin de ce segment d'une semaine en arrière, ce qui fait que les limites de date deviennent du 2 janvier au 7 janvier.

Diagramme de calendrier indiquant les dates de transaction qui renverraient un résultat booléen TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Par conséquent, toute transaction effectuée entre le 2 janvier et le 8 janvier (sans inclure le 8 janvier lui-même) renverra un résultat booléen `TRUE`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `in_week_to_date`, qui détermine les transactions qui ont eu lieu au cours de la semaine jusqu'au 14 janvier 2022.
- Création d'un champ supplémentaire appelé `weekday` via la fonction `weekday()`. Cela a pour objectif d'indiquer quel jour de la semaine correspond à chaque date.

Dans cet exemple, nous considérons le lundi comme le premier jour de la semaine.

Script de chargement

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0, 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- week_day
- in_week_to_date

Tableau de résultats

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Si on utilise 0 comme argument `first_week_day` dans la fonction `inweektodate()`, l'argument de la fonction remplace la variable système `FirstWeekDay` et définit le lundi comme le premier jour de la semaine.

Diagramme de calendrier indiquant les dates de transaction qui renverraient un résultat booléen TRUE

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	17
24	25	26	27	28	29	30
31						

Par conséquent, toute transaction qui se produit entre le 10 janvier et le 14 janvier renverra un résultat booléen TRUE, tandis que les transactions avec des dates en dehors de ces limites renverront une valeur FALSE.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu la même semaine jusqu'au 14 janvier est créé sous forme de mesure dans l'objet graphique.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
```

```
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.
2. Pour calculer si des transactions ont eu lieu au cours de la même semaine jusqu'au 14 janvier, créez la mesure suivante :
   `=inweektodate(date,'01/14/2022',0)`
3. Pour indiquer quel jour de la semaine correspond à chaque date, créez une mesure supplémentaire :
   `=weekday(date)`

Tableau de résultats

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0

date	week_day	in_week_to_date
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Le champ `in_week_to_date` est créé sous forme de mesure dans l'objet graphique à l'aide de la fonction `inweektodate()`. Le premier argument fourni identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 14 janvier, qui est la valeur `base_date` qui identifie la semaine à segmenter et définit la limite de fin du segment en question. Une valeur `period_no` égale à 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les semaines précédant ou suivant la semaine segmentée.

La variable système `FirstWeekDay` détermine que les semaines commencent un dimanche et se terminent un samedi. Par conséquent, janvier serait divisé en semaines selon le diagramme ci-dessous, les dates comprises entre le 9 et le 14 janvier fournissant la période valide pour le calcul `inweekdodate()` :

Diagramme de calendrier indiquant les dates de transaction qui renverraient un résultat booléen TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Toute transaction qui se produit entre le 9 janvier et le 14 janvier renvoie un résultat booléen TRUE. Les transactions avant et après les dates renvoient un résultat booléen FALSE.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Products`.
- Informations concernant l'ID du produit, la date de fabrication et le prix de revient.

Il a été identifié que suite à une erreur d'équipement, des produits fabriqués la semaine du 12 janvier étaient défectueux. Le problème a été résolu le 13 janvier. L'utilisateur final souhaite un objet graphique qui affiche, par semaine, l'état des produits fabriqués 'defective' (défectueux) ou 'faultless' (corrects) et le coût des produits fabriqués cette semaine-là.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau. Créez une dimension pour afficher les noms de semaine :
   `=weekname(manufacture_date)`
2. Ensuite, créez une dimension pour identifier les produits défectueux et les produits corrects :
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Créez une mesure pour additionner les valeurs `cost_price` des produits :
   `=sum(cost_price)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

weekname(manufacture_date)	if(inweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')	Sum(cost_price)
2022/02	Faultless	$200.09
2022/03	Defective	$263.46
2022/03	Faultless	$178.05
2022/04	Faultless	$178.41
2022/05	Faultless	$147.46
2022/06	Faultless	$248.12

La fonction `inweektodate()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Les produits qui renvoient une valeur booléenne `TRUE` sont identifiés comme `'Defective'`. Tout produit renvoyant une valeur `FALSE` et par conséquent non fabriqué au cours de la semaine jusqu'au 12 janvier est identifié comme `'Faultless'`.

inyear

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans l'année comprenant l'argument **base_date**.

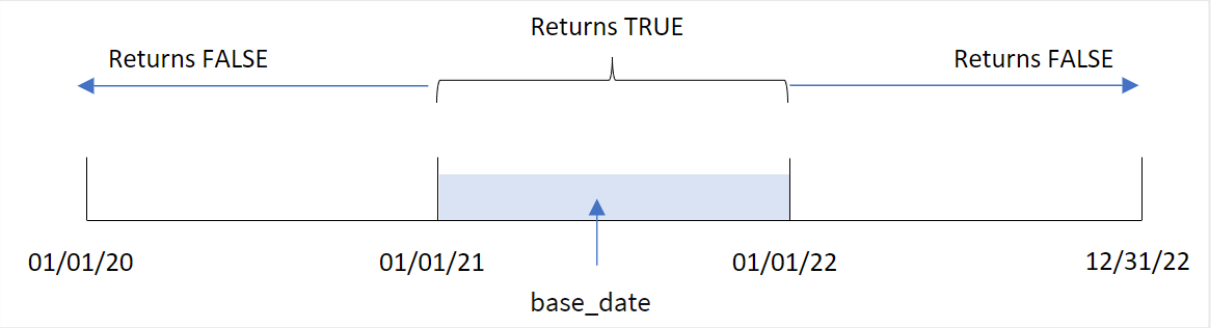
Syntaxe :

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Type de données renvoyé : booléen

Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.

Diagramme de la plage de la fonction `inyear()`



La fonction `inyear()` renvoie un résultat booléen lors de la comparaison des valeurs de date sélectionnées à une année définie par la `base_date`.

Cas d'utilisation

La fonction `inyear()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if expression`. Cela renvoie une agrégation ou un calcul suivant qu'une date évaluée s'est produite ou non au cours de l'année en question. Par exemple, la fonction `inyear()` permet d'identifier l'ensemble des ventes qui ont eu lieu au cours d'une année définie.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer l'année.
period_no	Il est possible de décaler l'année à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique l'année comprenant l'argument **base_date**. Les valeurs négatives de l'argument **period_no** indiquent les années passés tandis que les valeurs positives désignent les années à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5

Mois	Valeur
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`inyear ('01/25/2013', '01/01/2013', 0 )`	Renvoie TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Renvoie FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Renvoie FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	Renvoie TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Renvoie TRUE Les valeurs de `base_date` et de `first_month_of_year` spécifient que l'horodatage doit tomber entre le 01/03/2012 et le 02/28/2013.
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Renvoie TRUE

Exemple 1 - exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inyear()` définie comme le champ '`in_year`' et qui détermine les transactions qui ont eu lieu au cours de la même année que celle du 26 juillet 2021.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

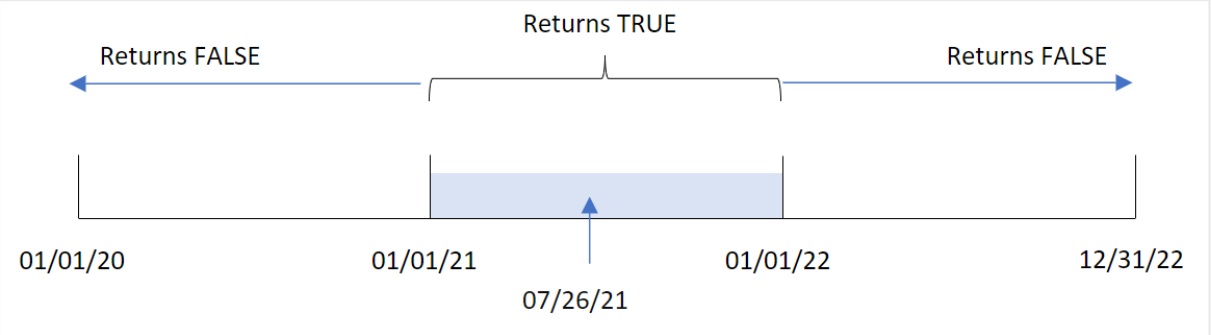
- `date`
- `in_year`

Tableau de résultats

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Le champ '`in_year`' est créé dans l'instruction LOAD précédente à l'aide de la fonction `inyear()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 26 juillet 2021, qui est la valeur `base_date` qui identifie l'année de comparaison. Un argument `period_no` égal à `0` constitue l'argument final, ce qui signifie que la fonction `inyear()` ne compare pas les années précédant ou suivant l'année.

Diagramme de la plage de la fonction `inyear()` avec le 26 juillet comme date de référence



Toute transaction qui a lieu en 2021 renvoie un résultat booléen TRUE.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inyear()` définie comme le champ '`previous_year`' et qui détermine les transactions qui ont eu lieu au cours de l'année précédant l'année contenant le 26 juillet 2021.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', -1) as previous_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
```

```
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_year`

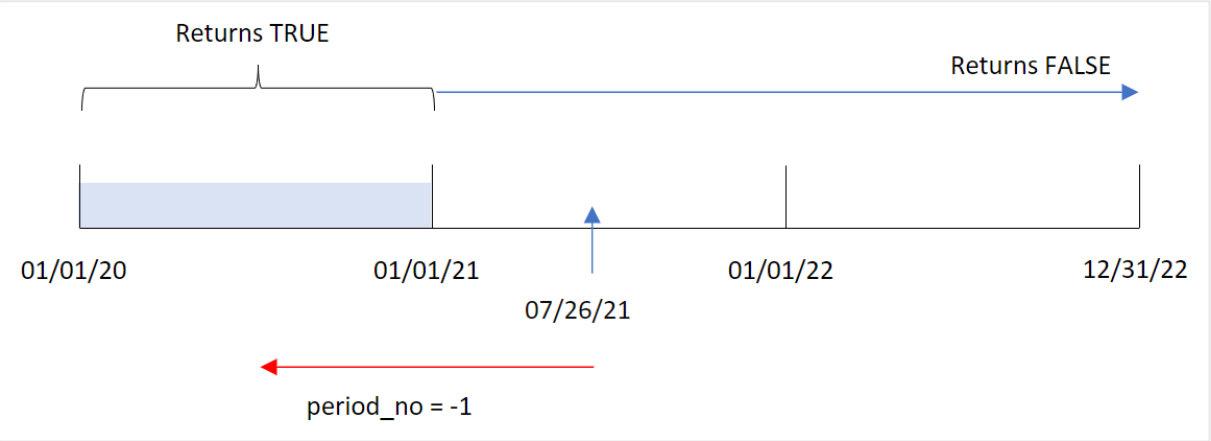
Tableau de résultats

date	previous_year
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
08/14/2020	-1
10/07/2020	-1
12/05/2020	-1
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
06/06/2022	0

date	previous_year
07/18/2022	0
11/14/2022	0
12/12/2022	0

Si on applique la valeur `-1` à l'argument `period_no` dans la fonction `inyear()`, les limites de l'année de comparaison sont décalées d'une année complète en arrière. L'année 2021 est initialement identifiée comme l'année de comparaison. L'argument`period_no` décale l'année de comparaison d'une année, faisant de l'année 2020 l'année de comparaison.

Diagramme de la plage de la fonction `inyear()` avec l'argument period_no défini sur `-1`



Par conséquent, toute transaction qui a lieu en 2020 renvoie un résultat booléen TRUE.

Exemple 3 - first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent qui contient la fonction `inyear()` définie comme le champ '`in_year`' et qui détermine les transactions qui ont eu lieu au cours de la même année que celle du 26 juillet 2021.

Cependant, dans cet exemple, la stratégie organisationnelle exige que mars soit le premier mois de l'exercice financier.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0, 3) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_year`

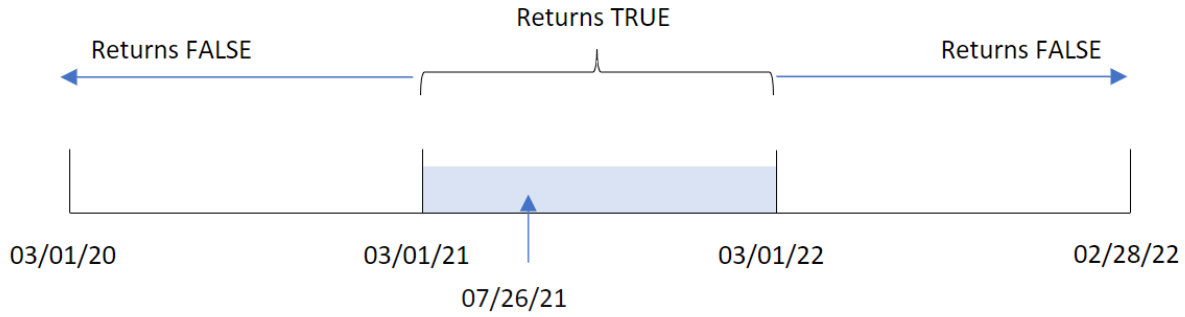
Tableau de résultats

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0

date	in_year
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

L'application de la valeur `3` à l'argument `first_month_of_year` dans la fonction `inyear()` fait commencer l'année le 1er mars et la fait se terminer à la fin du mois de février.

Diagramme de la plage de la fonction `inyear()` avec le mois de mars défini comme le premier mois de l'année



Par conséquent, toute transaction effectuée entre le 1er mars 2021 et le 1er mars 2022 renverra un résultat booléen TRUE.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui détermine si des transactions ont eu lieu au cours de la même année que celle du 26 juillet 2021 est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `date`

Pour calculer si des transactions ont eu lieu au cours de la même année que celle du 26 juillet 2021, créez la mesure suivante :

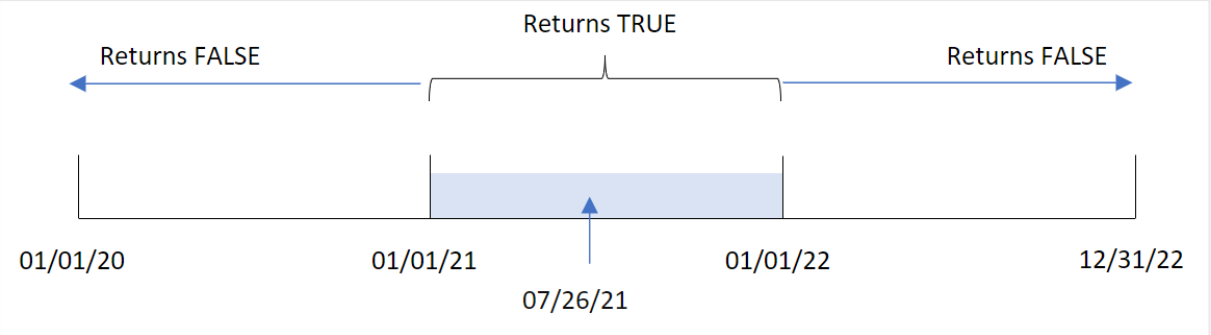
- =inyear(date,'07/26/2021', 0)

Tableau de résultats

date	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Le champ 'in_year' est créé dans le graphique à l'aide de la fonction `inyear()`. Le premier argument identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 26 juillet 2021, qui est la valeur `base_date` qui identifie l'année de comparaison. Un argument `period_no` égal à `0` constitue l'argument final, ce qui signifie que la fonction `inyear()` ne compare pas les années précédant ou suivant l'année.

Diagramme de la plage de la fonction `inyear()` avec le 27 juillet comme date de référence



Toute transaction qui a lieu en 2021 renvoie un résultat booléen TRUE.

Exemple 5 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Products`'.
- La table contient les champs suivants :
  - product ID
  - product type
  - manufacture date
  - cost price

L'utilisateur final souhaite un objet graphique qui affiche, par type de produit, le coût des produits fabriqués en 2021.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
```

```
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

- `product_type`

Pour calculer la somme de chaque produit fabriqué en 2021, créez la mesure suivante :

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Procédez comme suit :

1. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).
2. Sous **Aspect**, désactivez **Totals** (Totaux).

Tableau de résultats

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$171.21

La fonction `inyear()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué en 2021, la fonction `inyear()` renvoie une valeur booléenne TRUE et affiche la somme des valeurs `cost_price`.

inyeartodate

Cette fonction renvoie la valeur True si l'argument **timestamp** se trouve dans la partie de l'année comprenant l'argument **base_date** jusqu'à la dernière milliseconde spécifiée dans **base_date**.

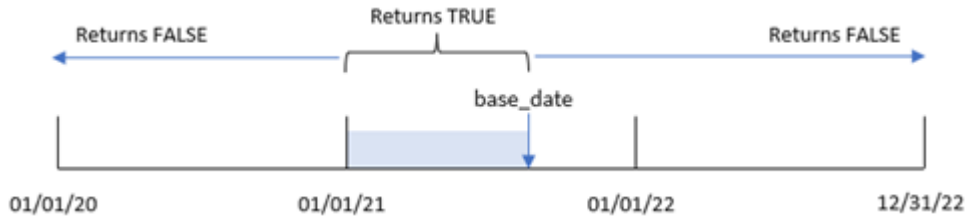
Syntaxe :

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Type de données renvoyé : booléen

> *Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.*

Diagramme de la fonction inyeartodate



La fonction `inyeartodate()` segmentera une portion donnée de l'année avec `base_date`, identifiant la date maximale autorisée pour ce segment d'année. La fonction évalue ensuite si un champ ou une valeur de date tombe dans ce segment et renvoie un résultat booléen.

Arguments

Argument	Description
timestamp	Date à comparer à **base_date**.
base_date	Date utilisée pour évaluer l'année.
period_no	Il est possible de décaler l'année à l'aide de l'argument **period_no**. **period_no** est un entier, où la valeur 0 indique l'année comprenant l'argument **base_date**. Les valeurs négatives de l'argument **period_no** indiquent les années passés tandis que les valeurs positives désignent les années à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `inyeartodate()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression `if`. Cela renverrait une agrégation ou un calcul suivant qu'une date évaluée s'est produite ou non au cours de l'année jusqu'à la date en question incluse.

Par exemple, la fonction `inyeartodate()` peut être utilisée pour identifier l'ensemble des équipements fabriqués au cours d'une année jusqu'à une date spécifique.

Ces exemples utilisent le format de date MM/DD/YYYY. Le format de date est indiqué dans l'instruction `SET DateFormat` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Renvoie `TRUE`.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Renvoie `FALSE`.
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Renvoie `TRUE`.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Renvoie `TRUE`. La valeur de `timestamp` tombe pendant l'exercice fiscal débutant au quatrième mois et avant la valeur définie par `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Renvoie `FALSE`. Par comparaison avec l'exemple précédent, la valeur de `timestamp` est toujours comprise dans l'exercice fiscal, mais elle se trouve après la valeur de `base_date`. Elle tombe donc en dehors de la partie de l'année spécifiée.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `in_year_to_date`, qui détermine les transactions qui ont eu lieu au cours de l'année jusqu'au 26 juillet 2021.

Script de chargement

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

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_year_to_date`

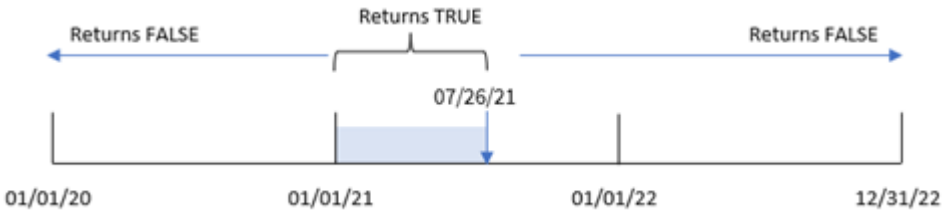
Tableau de résultats

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Le champ `in_year_to_date` est créé dans l'instruction LOAD précédente à l'aide de la fonction `inyeartodate()`. Le premier argument fourni identifie le champ en cours d'évaluation.

Le deuxième argument est une date codée en dur du 26 juillet 2021, qui est la valeur `base_date` qui identifie la limite de fin du segment d'année. Une valeur `period_no` égale à 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les années précédant ou suivant l'année segmentée.

Diagramme de la fonction inyeartodate, sans argument supplémentaire



Toute transaction qui se produit entre le 1er janvier et le 26 juillet renvoie un résultat booléen TRUE. Les dates de transactions avant 2021 et au-delà du 26 juillet 2021 renvoient FALSE.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_year_to_date`, qui détermine les transactions qui ont eu lieu une année complète avant la fin du segment d'année le 26 juillet 2021.

Script de chargement

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

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', -1) as previous_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
```

```
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_year_to_date`

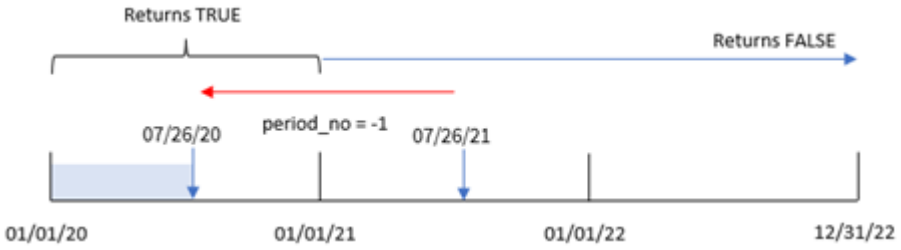
Tableau de résultats

date	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
06/14/2020	-1
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
07/27/2021	0
06/06/2022	0
07/18/2022	0

date	previous_year_to_date
11/14/2022	0
12/12/2022	0

Une valeur `period_no` égale à -1 indique que la fonction `inyeartodate ()` compare le segment de trimestre d'entrée à l'année précédente. Avec une date d'entrée du 26 juillet 2021, le segment du 1er janvier 2021 au 26 juillet 2021 a été initialement identifié comme le résultat de l'année jusqu'à la date en cours. La valeur `period_no` décale ensuite ce segment d'une année complète plus tôt, faisant passer les limites de date à la période du 1er janvier au 26 juillet 2020.

Diagramme de la fonction inyeartodate, exemple period_no



Par conséquent, toute transaction effectuée entre le 1er janvier et le 26 juillet 2020 renverra un résultat booléen `TRUE`.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `in_year_to_date`, qui détermine les transactions qui ont eu lieu au cours de la même année jusqu'au 26 juillet 2021.

Dans cet exemple, nous définissons mars comme le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0,3) as in_year_to_date
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `in_year_to_date`

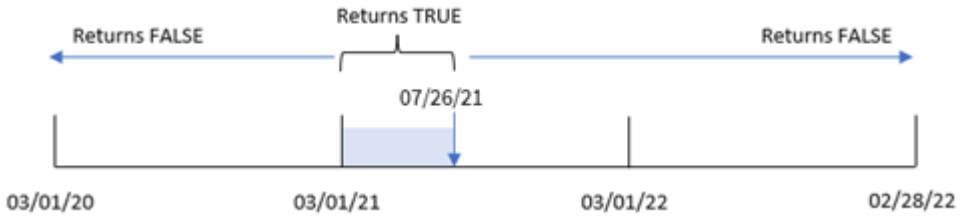
Tableau de résultats

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	0

date	in_year_to_date
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Si on utilise 3 comme argument `first_month_of_year` dans la fonction `inyeartodate()`, la fonction commence l'année le 1er mars. La valeur `base_date` du 26 juillet 2021 définit ensuite la date de fin de ce segment d'année-là.

Diagramme de la fonction inyeartodate, exemple first_month_of_year



Par conséquent, toute transaction qui se produit entre le 1er mars et le 26 juillet 2021 renverra un résultat booléen TRUE, tandis que les transactions avec des dates en dehors de ces limites renverront une valeur FALSE.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine si les transactions ont eu lieu la même année jusqu'au 26 juillet 2021 est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.

Créez la mesure suivante :

```
=inyeartodate(date,'07/26/2021', 0)
```

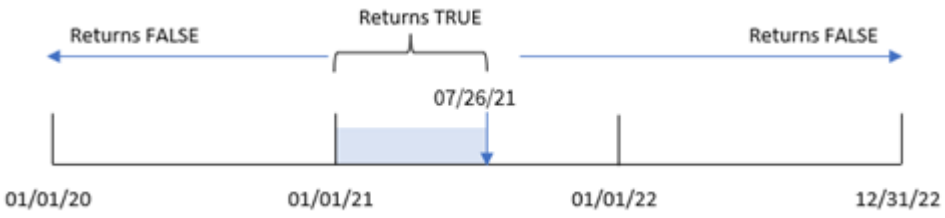
Tableau des résultats

date	=inyeartodate(date,'07/26/2021', 0)
01/13/2020	0
02/26/2020	0

date	=inyeartodate(date,'07/26/2021', 0)
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

La mesure `in_year_to_date` est créée dans l'objet graphique à l'aide de la fonction `inyeartodate()`. Le premier argument fourni identifie le champ en cours d'évaluation. Le deuxième argument est une date codée en dur du 26 juillet 2021, qui est la valeur `base_date` qui identifie la limite de fin du segment d'année du comparateur. Une valeur `period_no` égale à 0 constitue l'argument final, ce qui signifie que la fonction ne compare pas les années précédant ou suivant l'année segmentée.

Diagramme de la fonction inyeartodate, exemple objet graphique



Toute transaction qui se produit entre le 1er janvier et le 26 juillet 2021 renvoie un résultat booléen TRUE. Les dates de transactions avant 2021 et après le 26 juillet 2021 renvoient FALSE.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Products`.
- Informations concernant l'ID du produit, le type de produit, la date de fabrication et le prix de revient.

L'utilisateur final souhaite un objet graphique qui affiche, par type de produit, le coût des produits fabriqués en 2021 jusqu'au 26 juillet.

Script de chargement

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`product_type`.

Créez une mesure qui calcule la somme de chaque produit fabriqué en 2021 avant le 27 juillet :

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

product_type	=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$146.09

La fonction `inyeartodate()` renvoie une valeur booléenne lors de l'évaluation des dates de fabrication de chacun des produits. Pour tout produit fabriqué en 2021 avant le 27 juillet, la fonction `inyeartodate()` renvoie une valeur booléenne `TRUE` et additionne la valeur `cost_price`.

Product D est le seul produit également fabriqué après le 26 juillet 2021. L'entrée `product_ID` 8203 a été fabriquée le 27 décembre et a coûté $25.12. Par conséquent, ce coût n'a pas été inclus dans le total de Product D dans l'objet graphique.

lastworkdate

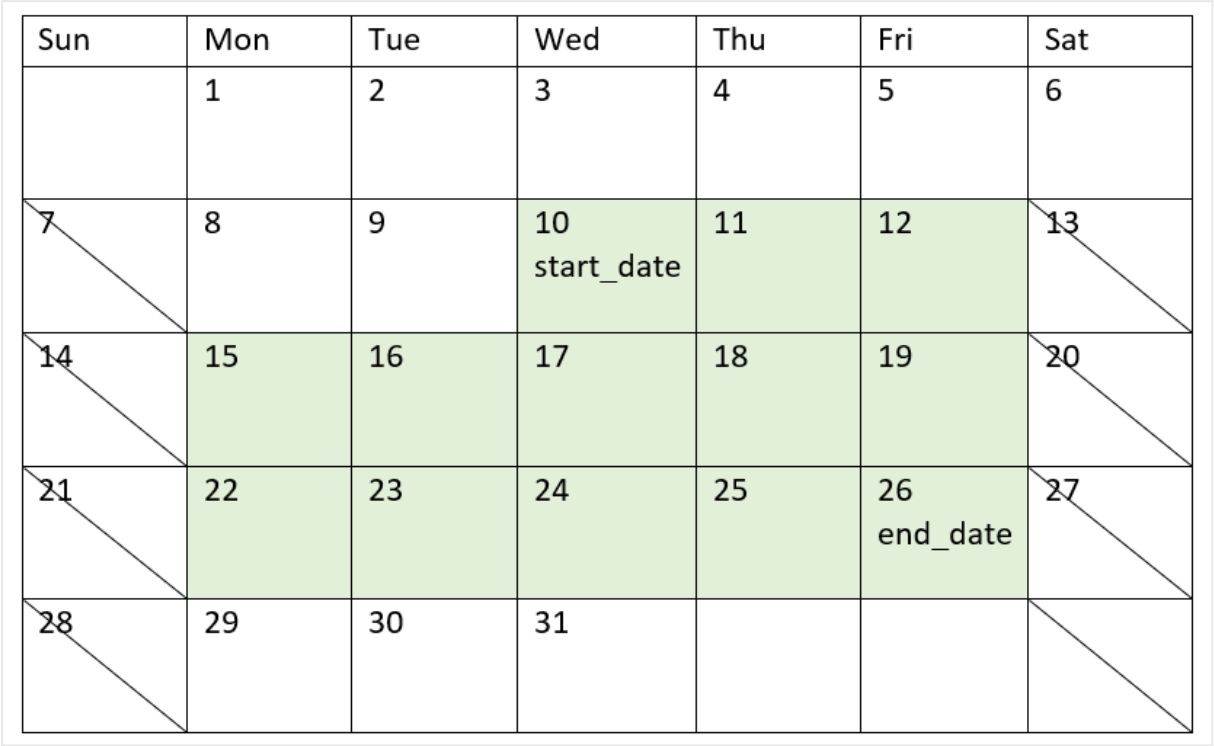
La fonction **lastworkdate** renvoie la première date de fin permettant d'atteindre la valeur de l'argument **no_of_workdays** (du lundi au vendredi) si celle-ci commence à la date définie par **start_date** en tenant compte de tous les arguments **holiday** facultatifs répertoriés. Les valeurs des arguments **start_date** et **holiday** doivent correspondre à des dates ou à des horodatages valides.

Syntaxe :

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Type de données renvoyé : entier

Calendrier indiquant comment utiliser la fonction `lastworkdate()`



Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Limitations

Il n'existe aucune méthode permettant de modifier la fonction `lastworkdate()` pour des régions ou des scénarios impliquant autre chose qu'une semaine de travail qui commence le lundi et se termine le vendredi.

Le paramètre holiday doit être une constante de type chaîne. Il n'accepte pas d'expression.

Cas d'utilisation

La fonction `lastworkdate()` est généralement utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite calculer la date de fin proposée d'un projet ou d'une mission, en fonction de la date de début du projet et des congés qui tomberont au cours de cette période.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Arguments

Argument	Description
start_date	Date de début à évaluer.
no_of_ workdays	Nombre de jours ouvrables à atteindre.
holiday	Périodes de congé à exclure des jours ouvrables. Un congé est indiqué sous forme de date constante de type chaîne. Vous pouvez spécifier plusieurs dates de congé si vous les séparez par des virgules. **Exemple :** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exemple 1 - exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant les ID de projet, les dates de début des projets et l'effort estimé, en jours, nécessaire pour les projets. L'ensemble de données est chargé dans une table appelée '`Projects`'.
- Chargement précédent contenant la fonction `lastworkdate()` définie comme le champ '`end_date`' et qui identifie la date de fin planifiée de chaque projet.

Script de chargement

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

Projects:
    Load
       *,
       LastWorkDate(start_date,effort) as end_date
       ;
Load
id,
start_date,
effort
```

```
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `effort`
- `end_date`

Tableau de résultats

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Étant donné qu'il n'existe aucun congé prévu, la fonction ajoute le nombre défini de jours ouvrables, du lundi au vendredi, à la date de début pour rechercher la date de fin la plus tôt possible.

Le calendrier suivant indique les dates de début et de fin du projet 3, avec les jours ouvrables surlignés en vert.

Calendrier indiquant les dates de début et de fin du projet 3

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19	20	21
22	23 End Date	24	25	26	27	28
29	30	31				

Exemple 2 - un seul jour de congé

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant les ID de projet, les dates de début des projets et l'effort estimé, en jours, nécessaire pour les projets. L'ensemble de données est chargé dans une table appelée '`Projects`'.
- Chargement précédent contenant la fonction `lastworkdate()` définie comme le champ '`end_date`' et qui identifie la date de fin planifiée de chaque projet.

Cependant, un jour de congé est prévu le 18 mai 2022. La fonction `lastworkdate()` du chargement précédent inclut le congé dans son troisième argument pour identifier la date de fin prévue de chaque projet.

Script de chargement

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

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/18/2022') as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- start_date
- effort
- end_date

Tableau de résultats

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/24/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Le seul jour de congé prévu est saisi comme troisième argument dans la fonction `lastworkdate()`. En conséquence, la date de fin du projet 3 est décalée d'un jour plus tard, car le jour de congé tombe un jour ouvrable avant la date de fin.

Le calendrier suivant indique les dates de début et de fin du projet 3 et montre que le jour de congé reporte la date de fin du projet d'un jour.

Calendrier indiquant les dates de début et de fin du projet 3 avec un jour de congé le 18 mai

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

Exemple 3 - plusieurs jours de congé

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant les ID de projet, les dates de début des projets et l'effort estimé, en jours, nécessaire pour les projets. L'ensemble de données est chargé dans une table appelée '`Projects`'.
- Chargement précédent contenant la fonction `lastworkdate()` définie comme le champ '`end_date`' et qui identifie la date de fin prévue de chaque projet.

Cependant, quatre jours de congé sont prévus les 19, 20, 21 et 22 mai. La fonction `lastworkdate()` du chargement précédent inclut chacun des jours de congé dans son troisième argument pour identifier la date de fin prévue de chaque projet.

Script de chargement

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

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/19/2022','05/20/2022','05/21/2022','05/22/2022') as
end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `effort`
- `end_date`

Tableau de résultats

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/25/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Les quatre jours de congé sont saisis sous la forme d'une liste d'arguments dans la fonction `lastworkdate ()` après la date de début et le nombre de jours ouvrables.

Le calendrier suivant indique les dates de début et de fin du projet 3 et montre que les jours de congé reportent la date de fin du projet de trois jours.

Calendrier indiquant les dates de début et de fin du projet 3 avec des jours de congé du 19 au 22 mai

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19 Holiday	20 Holiday	21 Holiday
22 Holiday	23	24	25 End Date	26	27	28
29	30	31				

Exemple 4 - un seul jour de congé (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est inchangé et chargé dans l'application. Le champ `end_date` est calculé sous forme de mesure dans un graphique.

Script de chargement

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

Projects:
Load
id,
start_date,
effort
Inline
[
```

```
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `effort`

Pour calculer la valeur end_date, créez la mesure suivante :

- `=LastWorkDate(start_date,effort,'05/18/2022')`

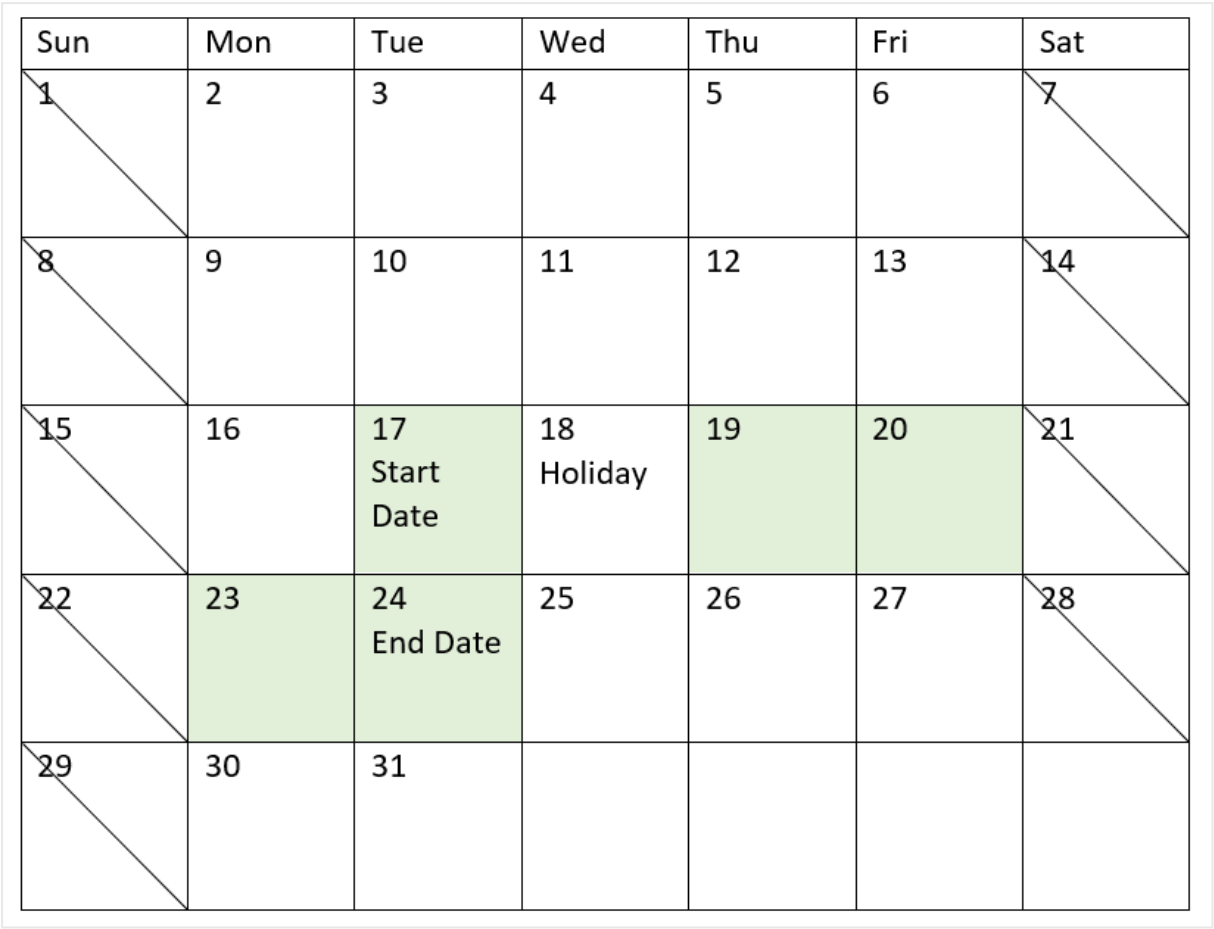
Tableau de résultats

id	start_date	effort	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Le seul jour de congé prévu est saisi comme mesure dans le graphique. En conséquence, la date de fin du projet 3 est reportée d'un jour, car le jour de congé tombe un jour ouvrable avant la date de fin.

Le calendrier suivant indique les dates de début et de fin du projet 3 et montre que le jour de congé reporte la date de fin du projet d'un jour.

Calendrier indiquant les dates de début et de fin du projet 3 avec un jour de congé le 18 mai



Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

localtime

Cette fonction renvoie un horodatage de l'heure actuelle pour un fuseau horaire donné.

Syntaxe :

```
LocalTime([timezone [, ignoreDST ]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
timezone	L'argument **timezone** est spécifié sous la forme d'une chaîne contenant l'un des lieux géographiques figurant sous **Fuseau horaire** dans le **Panneau de configuration de Windows** pour les **Date et heure** ou sous la forme d'une chaîne au format « GMT+hh:mm ». Si aucun fuseau horaire n'est spécifié, l'heure locale est renvoyée.
ignoreDST	Si l'argument **ignoreDST** est défini sur -1 (True), l'heure d'été est ignorée.

Exemples et résultats :

Les exemples ci-dessous sont basés sur la fonction appelée le 22 octobre 2014 (2014-10-22), à 12:54:47 heure locale, avec le fuseau horaire local GMT+01:00.

Exemples de script

Exemple	Résultat
`localtime ()`	Renvoie l'heure locale, soit 2014-10-22 12:54:47.
`localtime ('London')`	Renvoie l'heure locale de Londres, soit 2014-10-22 11:54:47.
`localtime ('GMT+02:00')`	Renvoie l'heure locale du fuseau horaire GMT+02:00, soit 2014-10-22 13:54:47.
`localtime ('Paris','-1')`	Renvoie l'heure locale de Paris sans tenir compte de l'heure d'été, soit 2014-10-22 11:54:47.

lunarweekend

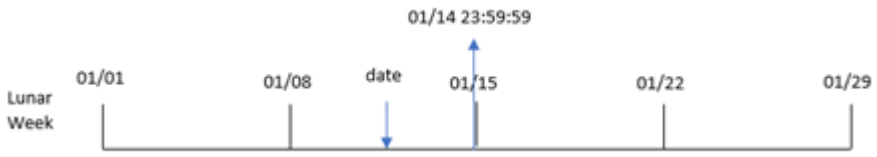
Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

Syntaxe :

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Type de données renvoyé : double

Exemple de diagramme de la fonction `lunarweekend()`



La fonction `lunarweekend()` détermine la semaine lunaire de la `date`. Elle renvoie ensuite un horodatage, au format date, pour la dernière milliseconde de cette semaine-là.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.

Argument	Description
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique la semaine lunaire contenant l'argument **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines lunaires passées tandis que les valeurs positives désignent les semaines lunaires à venir.
first_week_day	Décalage pouvant être supérieur ou inférieur à zéro. Il modifie le début de l'année du nombre de jours et/ou de fractions de jour spécifié.

Cas d'utilisation

La fonction `lunarweekend()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction de la semaine qui n'a pas encore eu lieu. Contrairement à la fonction `weekend()`, la dernière semaine lunaire de chaque année civile se termine le 31 décembre. Par exemple, la fonction `lunarweekend()` permet de calculer les intérêts non encore encourus au cours de la semaine.

Exemples de fonction

Exemple	Résultat
`lunarweekend('01/12/2013')`	Renvoie 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Renvoie 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Renvoie 01/15/2013 23:59:59.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `end_of_week`, qui renvoie un horodatage de la fin de la semaine lunaire au cours de laquelle les transactions ont eu lieu.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekend(date) as end_of_week,
        timestamp(lunarweekend(date)) as end_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `end_of_week`
- `end_of_week_timestamp`

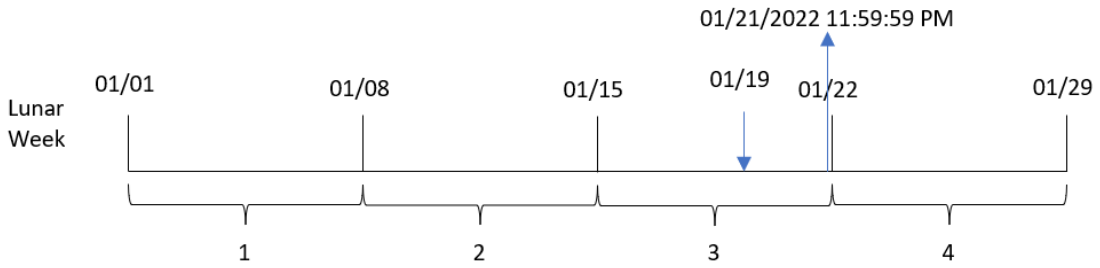
Tableau de résultats

date	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

Le champ `end_of_week` est créé dans l'instruction LOAD précédente via la fonction `lunarweekend()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `lunarweekend()` identifie la semaine lunaire de la valeur date, renvoyant un horodatage pour la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `lunarweekend()`, exemple sans argument supplémentaire



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekend()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la valeur `end_of_week` de cette transaction renvoie la dernière milliseconde de la semaine lunaire, à savoir, le 21 janvier à 11:59:59 PM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_lunar_week_end`, qui renvoie l'horodatage de la fin de la semaine lunaire avant la réalisation de la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekend(date,-1) as previous_lunar_week_end,
        timestamp(lunarweekend(date,-1)) as previous_lunar_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
```

```
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_lunar_week_end`
- `previous_lunar_week_end_timestamp`

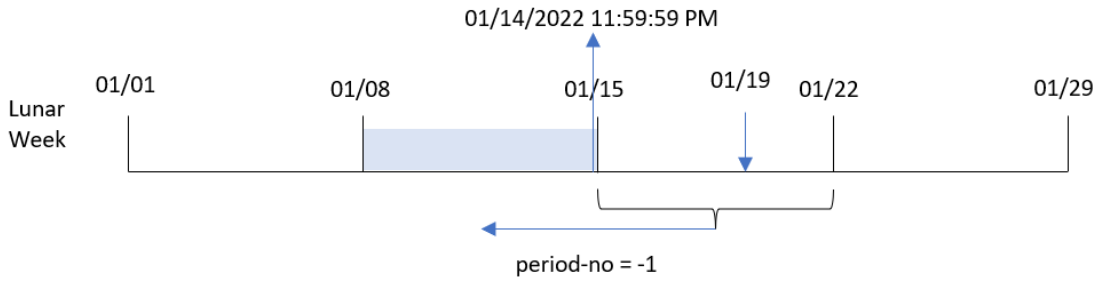
Tableau de résultats

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM
8/8/2022	08/05/2022	8/5/2022 11:59:59 PM
8/19/2022	08/12/2022	8/12/2022 11:59:59 PM
9/26/2022	09/23/2022	9/23/2022 11:59:59 PM
10/14/2022	10/07/2022	10/7/2022 11:59:59 PM
10/29/2022	10/28/2022	10/28/2022 11:59:59 PM

Dans cet exemple, étant donné qu'un argument `period_no` égal à -1 a été utilisé comme argument de décalage dans la fonction `lunarweekend()`, la fonction commence par identifier la semaine lunaire au cours de laquelle les transactions ont eu lieu. Elle décale ensuite d'une semaine en arrière et identifie la dernière milliseconde de cette semaine lunaire-là.

Diagramme de la fonction `lunarweekend()`, exemple period_no



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekend()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la semaine lunaire précédente a commencé le 8 janvier et s'est terminée le 14 janvier à 11:59:59 PM ; il s'agit de la valeur renvoyée pour le champ `previous_lunar_week_end`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Dans cet exemple, nous définissons les semaines lunaires de sorte qu'elles commencent le 5 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekend(date,0,4) as end_of_week,
timestamp(lunarweekend(date,0,4)) as end_of_week_timestamp
;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- end_of_week
- end_of_week_timestamp

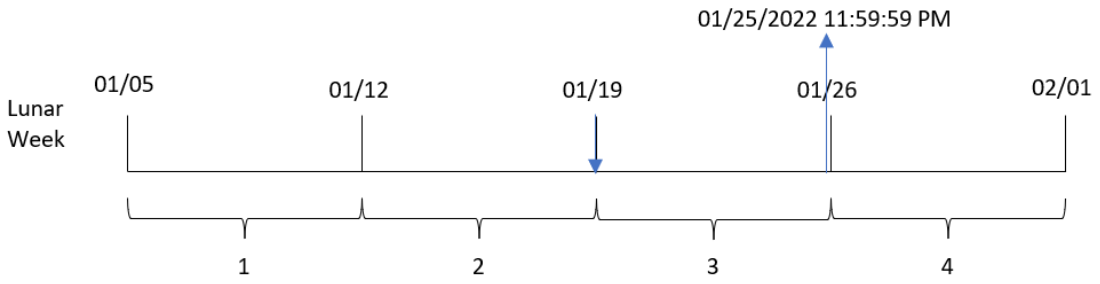
Tableau de résultats

date	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/11/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM
6/15/2022	06/21/2022	6/21/2022 11:59:59 PM
6/26/2022	06/28/2022	6/28/2022 11:59:59 PM
7/9/2022	07/12/2022	7/12/2022 11:59:59 PM
7/22/2022	07/26/2022	7/26/2022 11:59:59 PM
7/23/2022	07/26/2022	7/26/2022 11:59:59 PM
7/27/2022	08/02/2022	8/2/2022 11:59:59 PM
8/2/2022	08/02/2022	8/2/2022 11:59:59 PM
8/8/2022	08/09/2022	8/9/2022 11:59:59 PM
8/19/2022	08/23/2022	8/23/2022 11:59:59 PM
9/26/2022	09/27/2022	9/27/2022 11:59:59 PM
10/14/2022	10/18/2022	10/18/2022 11:59:59 PM
10/29/2022	11/01/2022	11/1/2022 11:59:59 PM

Dans cet exemple, étant donné que l'argument `first_week_date` égal à 4 est utilisé dans la fonction `lunarweekend()`, il reporte le début de l'année du 1er janvier au 5 janvier.

Diagramme de la fonction `lunarweekend()`, exemple first_week_day



La transaction 8189 a eu lieu le 19 janvier. Étant donné que les semaines lunaires commencent le 5 janvier, la fonction `lunarweekend()` identifie que la semaine lunaire contenant le 19 janvier commence également le 19 janvier. Par conséquent, la fin de cette semaine lunaire a lieu le 25 janvier à 11:59:59 PM ; il s'agit de la valeur renvoyée pour le champ `end_of_week`.

Exemple 4 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage pour la fin de la semaine lunaire des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Ajoutez les mesures suivantes :

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

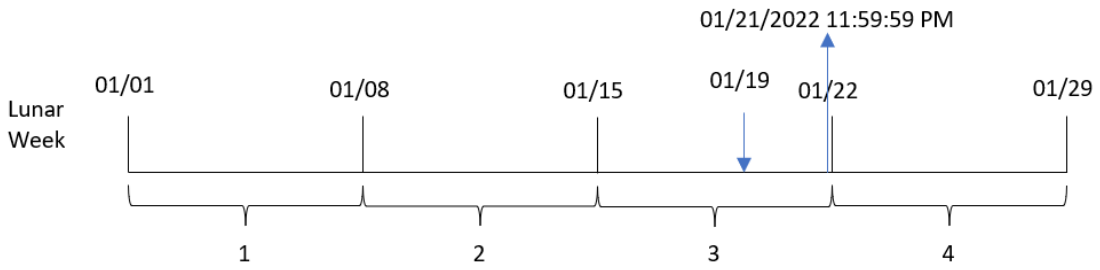
Tableau de résultats

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

La mesure `end_of_week` est créée dans l'objet graphique via la fonction `lunarweekend()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `lunarweekend()` identifie la semaine lunaire de la valeur date, renvoyant un horodatage pour la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `lunarweekend()`, exemple objet graphique



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekend()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la valeur `end_of_week` de cette transaction renvoie la dernière milliseconde de la semaine lunaire, à savoir, le 21 janvier à 11:59:59 PM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Employee_Expenses`.
- ID des employés, nom des employés et notes de frais quotidiennes moyennes de chaque employé.

L'utilisateur final souhaite un objet graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées qu'il reste à encourir pour le reste de la semaine lunaire.

Script de chargement

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Ajoutez les champs suivants comme dimensions :
   - `employee_id`
   - `employee_name`
3. Ensuite, pour calculer les intérêts cumulés, créez la mesure suivante :
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
182	Mark	$75.00
183	Deryck	$62.50
184	Dexter	$62.50
185	Sydney	$135.00
186	Agatha	$90.00

La fonction `lunarkweekend()`, en utilisant la date d'aujourd'hui comme son seul argument, renvoie la date de fin de la semaine lunaire en cours. Ensuite, en soustrayant la date d'aujourd'hui de la date de fin de la semaine lunaire, l'expression renvoie le nombre de jours restants pour cette semaine.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes par employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours du reste de la semaine lunaire.

lunarweekname

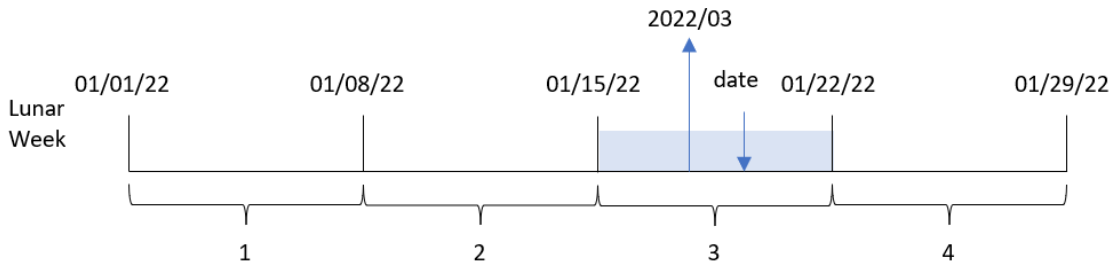
Cette fonction renvoie une valeur d'affichage indiquant l'année et le numéro de la semaine lunaire correspondant à un horodatage de la première milliseconde du premier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

Syntaxe :

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Type de données renvoyé : double

Exemple de diagramme de la fonction `lunarweekname()`



La fonction `lunarweekname()` détermine la semaine lunaire de la date, en commençant le comptage des semaines le 1er janvier. Elle renvoie ensuite une valeur comprenant `year/weekcount`.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique la semaine lunaire contenant l'argument **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines lunaires passées tandis que les valeurs positives désignent les semaines lunaires à venir.
first_week_day	Décalage pouvant être supérieur ou inférieur à zéro. Il modifie le début de l'année du nombre de jours et/ou de fractions de jour spécifié.

Cas d'utilisation

La fonction `lunarweekname()` est utile lorsque vous souhaitez comparer des agrégations par semaines lunaires. Par exemple, la fonction pourrait permettre de déterminer les ventes totales de produits par semaine lunaire. Les semaines lunaires s'avèrent utiles lorsque vous souhaitez vous assurer que toutes les valeurs contenues dans la première semaine de l'année correspondent uniquement à des valeurs à partir du 1er janvier au plus tôt.

Il est possible de créer ces dimensions dans le script de chargement via la fonction permettant de créer un champ dans une table Calendrier principal. La fonction peut également être utilisée directement dans un graphique comme dimension calculée.

Exemples de fonction

Exemple	Résultat
`lunarweekname('01/12/2013')`	Renvoie 2006/02.
`lunarweekname('01/12/2013', -1)`	Renvoie 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Renvoie 2006/02.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – date sans aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `lunar_week_name`, qui renvoie l'année et le numéro de semaine de la semaine lunaire au cours de laquelle les transactions ont eu lieu.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekname(date) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
```

```
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `lunar_week_name`

Tableau de résultats

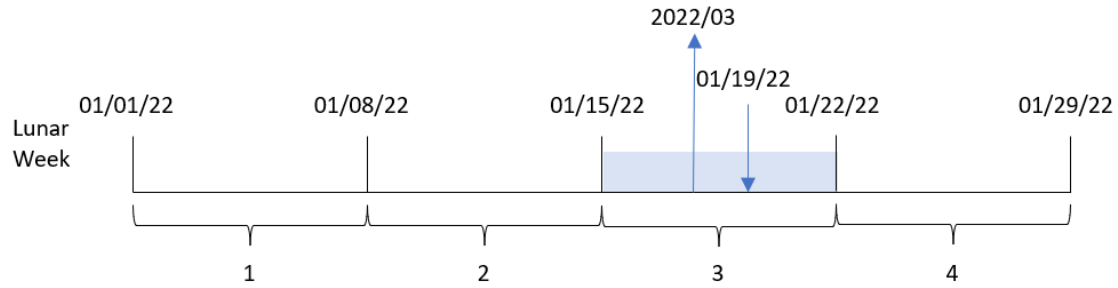
date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30

date	lunar_week_name
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

Le champ `lunar_week_name` est créé dans l'instruction LOAD précédente via la fonction `lunarweekname()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `lunarweekname()` identifie la semaine lunaire de la valeur date en renvoyant l'année et le numéro de semaine de cette date.

Diagramme de la fonction `lunarweekname()`, exemple sans argument supplémentaire



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekname()` identifie que cette date tombe pendant la semaine lunaire commençant le 15 janvier ; il s'agit de la troisième semaine lunaire de l'année. Par conséquent, la valeur `lunar_week_name` renvoyée pour cette transaction est 2022/03.

Exemple 2 – date avec argument period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_lunar_week_name`, qui renvoie l'année et le numéro de semaine de la semaine lunaire précédant les transactions.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekname(date,-1) as previous_lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_lunar_week_name`

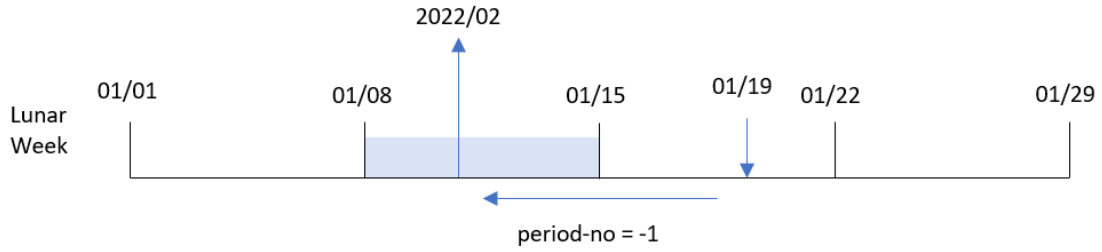
Tableau de résultats

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08

date	previous_lunar_week_name
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

Dans cet exemple, étant donné qu'un argument `period_no` égal à -1 a été utilisé comme argument de décalage dans la fonction `lunarweekname()`, la fonction commence par identifier la semaine lunaire au cours de laquelle les transactions ont eu lieu. Elle renvoie ensuite l'année et le numéro d'une semaine avant.

Diagramme de la fonction `lunarweekname()`, exemple period_no



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekname()` identifie que cette transaction a eu lieu au cours de la troisième semaine lunaire de l'année ; c'est pourquoi elle renvoie l'année et la valeur d'une semaine auparavant, 2022/02, pour le champ `previous_lunar_week_name`.

Exemple 3 – date avec argument first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Dans cet exemple, nous définissons les semaines lunaires de sorte qu'elles commencent le 5 janvier.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekname(date,0,4) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

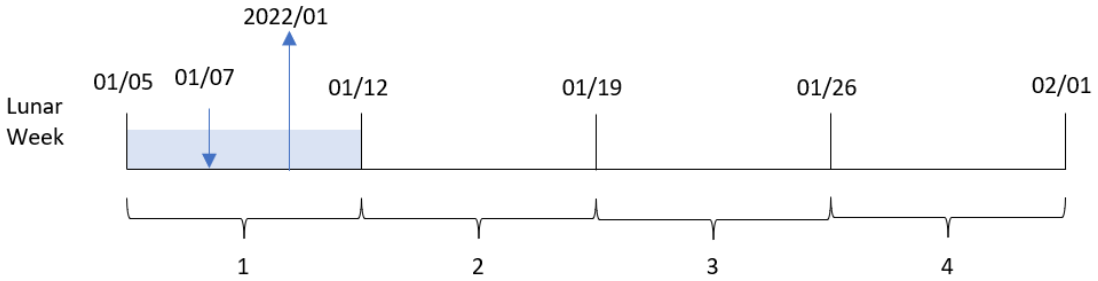
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `lunar_week_name`

Tableau de résultats

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/29
7/23/2022	2022/29
7/27/2022	2022/30
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/33
9/26/2022	2022/38
10/14/2022	2022/41
10/29/2022	2022/43

Diagramme de la fonction `lunarweekname()`, exemple first_week_day



Dans cet exemple, étant donné que l'argument `first_week_date` égal à 4 est utilisé dans la fonction `lunarweekname()`, il reporte le début des semaines lunaires du 1er janvier au 5 janvier.

La transaction 8188 a eu lieu le 7 janvier. Étant donné que les semaines lunaires commencent le 5 janvier, la fonction `lunarweekname()` identifie que la semaine lunaire contenant le 7 janvier est la première semaine lunaire de l'année. Par conséquent, la valeur `lunar_week_name` renvoyée pour cette transaction est 2022/01.

Exemple 4 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie le numéro de semaine lunaire et l'année des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer la date de début de la semaine lunaire d'une transaction, créez la mesure suivante :

```
=lunarweekname(date)
```

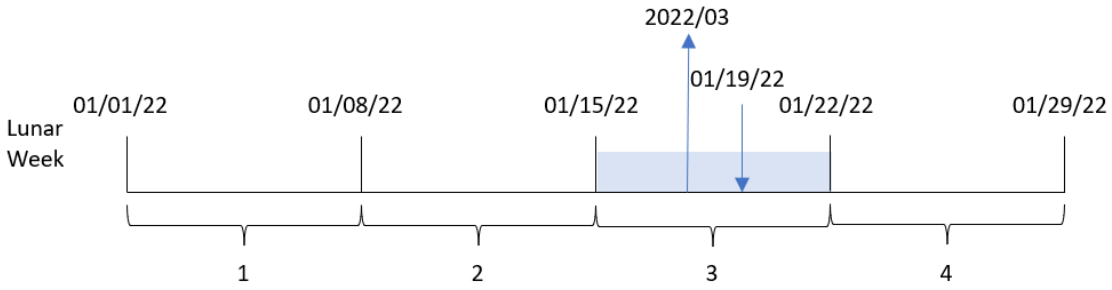
Tableau de résultats

date	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

La mesure `lunar_week_name` est créée dans l'objet graphique via la fonction `lunarweekname()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `lunarweekname()` identifie la semaine lunaire de la valeur date en renvoyant l'année et le numéro de semaine de cette date.

Diagramme de la fonction `lunarweekname()`, exemple objet graphique



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekname()` identifie que cette date tombe pendant la semaine lunaire commençant le 15 janvier ; il s'agit de la troisième semaine lunaire de l'année. Par conséquent, la valeur `lunar_week_name` de cette transaction est 2022/03.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

L'utilisateur final souhaite un objet graphique présentant les ventes totales par semaine pour l'année en cours. La semaine 1, d'une longueur de sept jours, doit commencer le 1er janvier. Il est possible d'y parvenir, même lorsque cette dimension n'est pas disponible dans le modèle de données, via la fonction `lunarweekname()` utilisée comme dimension calculée dans le graphique.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
```

```
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez une dimension calculée à l'aide de l'expression suivante :
   `=lunarweekname(date)`
3. Pour calculer les ventes totales, utilisez la mesure d'agrégation suivante :
   `=sum(amount)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66

=lunarweekname(date)	=sum(amount)
2022/30	$152.75
2022/31	$76.11
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour de la semaine lunaire contenant l'argument **date**. Dans Qlik Sense, les semaines lunaires sont définies en comptant le 1er janvier comme le premier jour de la semaine et, à l'exception de la dernière semaine de l'année, elles contiendront exactement sept jours.

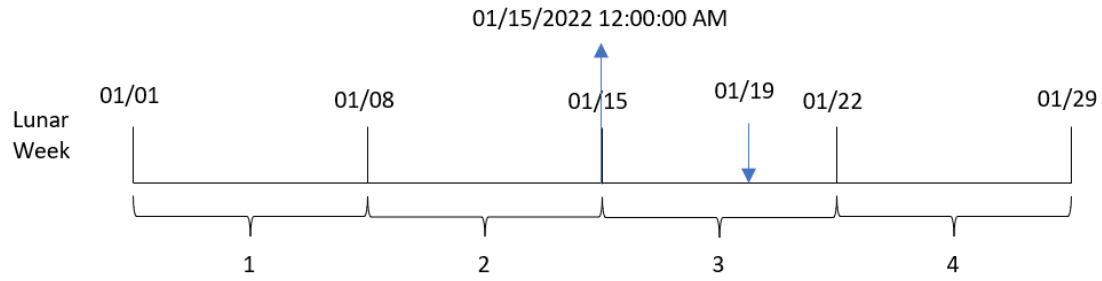
Syntaxe :

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Type de données renvoyé : double

La fonction `lunarweekstart()` détermine la semaine lunaire de la `date`. Elle renvoie ensuite un horodatage, au format date, pour la première milliseconde de cette semaine-là.

Exemple de diagramme de la fonction `lunarweekstart()`



Arguments

Argument	Description
date	Date ou horodatage à évaluer.

Argument	Description
period_no	**period_no** est un entier ou une expression qui aboutit à un entier, où la valeur 0 indique la semaine lunaire contenant l'argument **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les semaines lunaires passées tandis que les valeurs positives désignent les semaines lunaires à venir.
first_week_day	Décalage pouvant être supérieur ou inférieur à zéro. Il modifie le début de l'année du nombre de jours et/ou de fractions de jour spécifié.

Cas d'utilisation

La fonction `lunarweekstart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction de la semaine qui s'est écoulée jusqu'à présent. Contrairement à la fonction `weekstart()`, au début de chaque nouvelle année civile, la semaine commence le 1er janvier et chaque semaine suivante commence sept jours plus tard. La fonction`lunarweekstart()` n'est pas affectée par la variable système `FirstWeekDay`.

Par exemple, vous pouvez utiliser la fonction `lunarweekstart()` pour calculer les intérêts cumulés au cours d'une semaine jusqu'à la date d'aujourd'hui.

Exemples de fonction

Exemple	Résultat
`lunarweekstart ('01/12/2013')`	Renvoie 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Renvoie 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Renvoie 01/09/2013, car la définition de first_week_day sur 1 signifie que le début de l'année est devenu le 01/02/2013.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `start_of_week`, qui renvoie un horodatage du début de la semaine lunaire au cours de laquelle les transactions ont eu lieu.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekstart(date) as start_of_week,
        timestamp(lunarweekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `start_of_week`
- `start_of_week_timestamp`

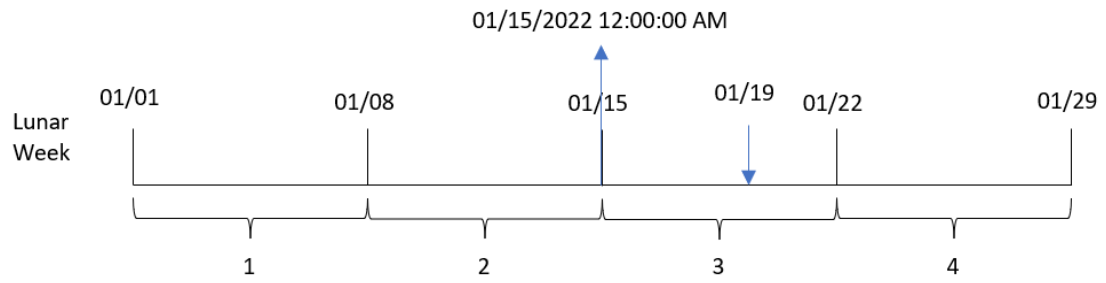
Tableau de résultats

date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

Le champ `start_of_week` est créé dans l'instruction LOAD précédente via la fonction `lunarweekstart()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `lunarweekstart()` identifie la semaine lunaire de la date en renvoyant un horodatage pour la première milliseconde de cette semaine-là.

Diagramme de la fonction `lunarweekstart()`, exemple sans argument supplémentaire



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekstart()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la valeur `start_of_week` de cette transaction renvoie la première milliseconde de ce jour-là, à savoir, le 15 janvier à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_lunar_week_start`, qui renvoie l'horodatage du début de la semaine lunaire avant la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        lunarweekstart(date,-1) as previous_lunar_week_start,
        timestamp(lunarweekstart(date,-1)) as previous_lunar_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
```

```
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

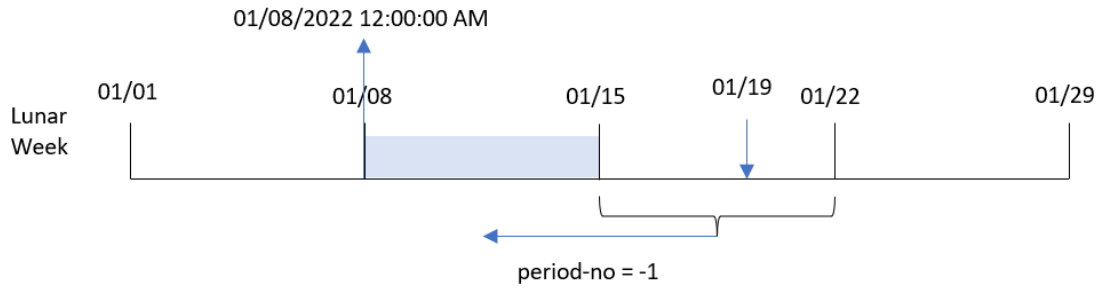
Tableau de résultats

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
1/7/2022	12/24/2021	12/24/2021 12:00:00 AM
1/19/2022	01/08/2022	1/8/2022 12:00:00 AM
2/5/2022	01/29/2022	1/29/2022 12:00:00 AM
2/28/2022	02/19/2022	2/19/2022 12:00:00 AM
3/16/2022	03/05/2022	3/5/2022 12:00:00 AM
4/1/2022	03/19/2022	3/19/2022 12:00:00 AM
5/7/2022	04/30/2022	4/30/2022 12:00:00 AM
5/16/2022	05/07/2022	5/7/2022 12:00:00 AM
6/15/2022	06/04/2022	6/4/2022 12:00:00 AM
6/26/2022	06/18/2022	6/18/2022 12:00:00 AM
7/9/2022	07/02/2022	7/2/2022 12:00:00 AM
7/22/2022	07/09/2022	7/9/2022 12:00:00 AM
7/23/2022	07/16/2022	7/16/2022 12:00:00 AM
7/27/2022	07/16/2022	7/16/2022 12:00:00 AM
8/2/2022	07/23/2022	7/23/2022 12:00:00 AM
8/8/2022	07/30/2022	7/30/2022 12:00:00 AM

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
8/19/2022	08/06/2022	8/6/2022 12:00:00 AM
9/26/2022	09/17/2022	9/17/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/22/2022	10/22/2022 12:00:00 AM

Dans cet exemple, étant donné que l'argument `period_no` égal à -1 a été utilisé comme argument de décalage dans la fonction `lunarweekstart()`, la fonction commence par identifier la semaine lunaire au cours de laquelle les transactions ont lieu. Elle décale ensuite d'une semaine en arrière et identifie la première milliseconde de cette semaine lunaire.

Diagramme de la fonction `lunarweekstart()`, exemple period_no



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekstart()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la semaine lunaire précédente a commencé le 8 janvier à 12:00:00 AM ; il s'agit de la valeur renvoyée pour le champ `previous_lunar_week_start`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Dans cet exemple, nous définissons les semaines lunaires de sorte qu'elles commencent le 5 janvier.

Script de chargement

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

Transactions:
    Load
        *,
```

```
        lunarweekstart(date,0,4) as start_of_week,
        timestamp(lunarweekstart(date,0,4)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- start_of_week
- start_of_week_timestamp

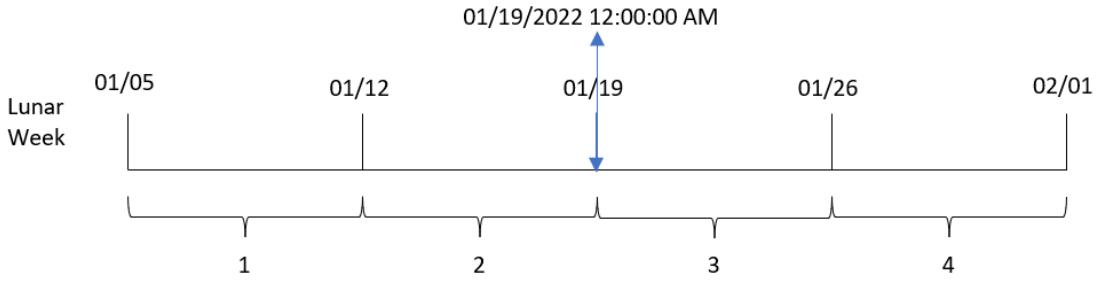
Tableau de résultats

date	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/2/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM
9/26/2022	09/21/2022	9/21/2022 12:00:00 AM
10/14/2022	10/12/2022	10/12/2022 12:00:00 AM
10/29/2022	10/26/2022	10/26/2022 12:00:00 AM

Dans cet exemple, étant donné que l'argument `first_week_date` égal à 4 est utilisé dans la fonction `lunarweekstart()`, il reporte le début de l'année du 1er janvier au 5 janvier.

Diagramme de la fonction `lunarweekstart()`, exemple first_week_day



La transaction 8189 a eu lieu le 19 janvier. Étant donné que les semaines lunaires commencent le 5 janvier, la fonction `lunarweekstart()` identifie que la semaine lunaire contenant le 19 janvier commence également le 19 janvier à 12:00:00 AM. Il s'agit par conséquent de la valeur renvoyée pour le champ `start_of_week`.

Exemple 4 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage pour le début de la semaine lunaire des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Ajoutez les mesures suivantes :

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

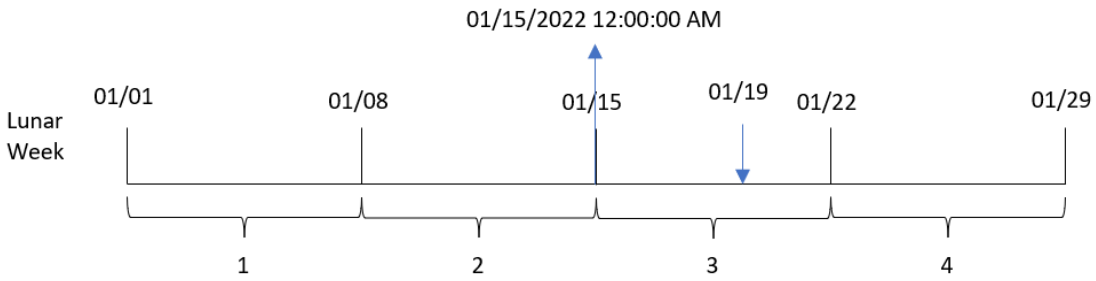
Tableau de résultats

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

La mesure `start_of_week` est créée dans l'objet graphique via la fonction `lunarweekstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `lunarweekstart()` identifie la semaine lunaire de la valeur date, renvoyant un horodatage pour la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `lunarweekstart()`, exemple objet graphique



La transaction 8189 a eu lieu le 19 janvier. La fonction `lunarweekstart()` identifie que la semaine lunaire commence le 15 janvier. Par conséquent, la valeur `start_of_week` de cette transaction est la première milliseconde de ce jour-là, à savoir, le 15 janvier à 12:00:00 AM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de soldes de prêts, chargé dans une table appelée `Loans`.
- Données constituées des ID des prêts, du solde au début de la semaine et du taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels cumulés pour chaque prêt au cours de la semaine jusqu'à la date du jour.

Script de chargement

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Ajoutez les champs suivants comme dimensions :
   - `loan_id`
   - `start_balance`
3. Ensuite, pour calculer les intérêts cumulés, créez la mesure suivante :
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)- lunarweekstart (today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Si on utilise la date d'aujourd'hui comme seul argument, la fonction `lunarweekstart()` renvoie la date de début de l'année en cours. En soustrayant ce résultat de la date actuelle, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent cette semaine.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif encouru pour cette période. Le résultat est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts cumulés jusqu'à présent cette semaine.

makedate

Cette fonction renvoie une date calculée à partir de l'année **YYYY**, du mois **MM** et du jour **DD**.

Syntaxe :

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Type de données renvoyé : double

Arguments

Argument	Description
YYYY	Année sous forme d'entier.

Argument	Description
MM	Mois sous forme d'entier. Si aucun mois n'est spécifié, 1 (janvier) est utilisé.
DD	Jour sous forme d'entier. Si aucun jour n'est spécifié, la fonction utilise 1 (le premier).

Cas d'utilisation

La fonction `makedate()` est généralement utilisée dans le script pour la génération de données afin de produire un calendrier. Cela peut également être utilisé lorsque le champ date n'est pas directement disponible sous forme de date, mais qu'il doit subir des transformations pour en extraire les composants year, (année) month (mois) et day (jour).

Ces exemples utilisent le format de date MM/DD/YYYY. Le format de date est indiqué dans l'instruction `SET DateFormat` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`makedate(2012)`	Renvoie 01/01/2012.
`makedate(12)`	Renvoie 01/01/2012.
`makedate(2012,12)`	Renvoie 12/01/2012.
`makedate(2012,2,14)`	Renvoie 02/14/2012.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2018, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `transaction_date`, qui renvoie une date au format MM/DD/YYYY.

Script de chargement

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

Transactions:
    Load
        *,
        makedate(transaction_year, transaction_month, transaction_day) as transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tableau de résultats

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	08/30/2018
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	09/23/2018

Le champ `transaction_date` est créé dans l'instruction LOAD précédente via la fonction `makedate()` et en transmettant les champs year, month et day comme arguments de la fonction.

La fonction combine ensuite ces valeurs et les convertit en un champ date, renvoyant les résultats au format de la variable système `DateFormat`.

Exemple 2 – DateFormat modifié

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `transaction_date`, au format DD/MM/YYYY, sans modifier la variable système `DateFormat`.

Script de chargement

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

Transactions:
    Load
        *,
        date(makedate(transaction_year, transaction_month, transaction_day), 'DD/MM/YYYY') as
transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
```

```
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tableau de résultats

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	30/08/2018
2018	09	07	07/09/2018
2018	09	16	16/09/2018
2018	09	22	22/09/2018
2018	09	23	23/09/2018

Dans cet exemple, la fonction `makedate()` est imbriquée dans la fonction `date()`. Le deuxième argument de la fonction `date()` définit le format des résultats de la fonction `makedate()` sur le format DD/MM/YYYY requis.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2018, chargé dans une table appelée `Transactions`.
- Dates de transaction fournies sur deux champs : `year` et `month`.

Créez une mesure d'objet graphique, `transaction_date`, qui renvoie une date au format MM/DD/YYYY.

Script de chargement

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

Transactions:
Load * Inline [
```

```
transaction_id, transaction_year, transaction_month, transaction_amount, transaction_quantity,
customer_id
3750, 2018, 08, 12423.56, 23, 2038593
3751, 2018, 09, 5356.31, 6, 203521
3752, 2018, 09, 15.75, 1, 5646471
3753, 2018, 09, 1251, 7, 3036491
3754, 2018, 09, 21484.21, 1356, 049681
3756, 2018, 09, -59.18, 2, 2038593
3757, 2018, 09, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- year
- month

Pour déterminer la valeur transaction_date, créez la mesure suivante :

```
=makedate(transaction_year,transaction_month)
```

Tableau de résultats

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

La mesure transaction_date est créée dans l'objet graphique via la fonction makedate() et en transmettant les champs year et month comme arguments de la fonction.

La fonction combine ensuite ces valeurs ainsi que la valeur day supposée égale à 01. Ces valeurs sont ensuite converties en un champ date, renvoyant les résultats au format de la variable système DateFormat.

Exemple 4 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Créez un ensemble de données Calendar pour l'année civile 2022.

Script de chargement

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

Calendar:
       load
                *
    where year(date)=2022;
load
```

```
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Résultats

Tableau de résultats

date
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022
01/22/2022
01/23/2022
01/24/2022
01/25/2022
+ 340 lignes supplémentaires

La fonction `makedate()` crée une valeur date pour le 31 décembre 2021. La fonction `recno()` fournit le numéro d'enregistrement de l'enregistrement en cours de chargement dans la table, en commençant par 1. Par conséquent, le premier enregistrement porte la date du 1er janvier 2022. Chaque valeur `recno()` successive incrémentera ensuite cette date de 1. Cette expression est imbriquée dans une fonction `date()` afin de convertir la valeur en date. La fonction `autogenerate` répète ce processus 400 fois. Pour finir, via un chargement précédent, il est possible d'utiliser une condition `where` pour charger uniquement les dates de l'année 2022. Ce script génère un calendrier contenant chaque date en 2022.

maketime

Cette fonction renvoie une heure calculée à partir de l'heure **hh**, de la minute **mm** et de la seconde **ss**.

Syntaxe :

```
MakeTime(hh [ , mm [ , ss ] ])
```

Type de données renvoyé : double

Arguments

Argument	Description
hh	Heure sous forme d'entier.
mm	Minute sous forme d'entier. Si aucune minute n'est spécifiée, 00 est utilisé.
ss	Seconde sous forme d'entier. Si aucune seconde n'est spécifiée, 00 est utilisé.

Cas d'utilisation

La fonction `maketime()` est généralement utilisée dans le script pour la génération de données afin de produire un champ time (heure). Parfois, lorsque le champ time est dérivé du texte d'entrée, cette fonction peut être utilisée pour obtenir l'heure via ses composants.

Ces exemples utilisent le format horaire `h:mm:ss`. Le format horaire est indiqué dans l'instruction `SET TimeFormat` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`maketime(22)`	Renvoie 22:00:00.
`maketime(22, 17)`	Renvoie 22:17:00.
`maketime(22,17,52 )`	Renvoie 22:17:52.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – maketime()

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions, chargé dans une table appelée `Transactions`.
- Heures de transaction fournies sur trois champs : `hours`, `minutes` et `seconds`.
- Création d'un champ, `transaction_time`, qui renvoie l'heure au format de la variable système `TimeFormat`.

Script de chargement

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        maketime(transaction_hour, transaction_minute, transaction_second) as transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
```

```
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tableau de résultats

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22 AM
6	32	07	6:32:07 AM
9	25	23	9:25:23 AM
12	09	16	12:09:16 PM
17	55	22	5:55:22 PM
18	43	30	6:43:30 PM
21	43	41	9:43:41 PM

Le champ `transaction_time` est créé dans l'instruction LOAD précédente via la fonction `maketime()` et en transmettant les champs hour, minute et second comme arguments de la fonction.

La fonction combine ensuite ces valeurs et les convertit en un champ time, renvoyant les résultats au format horaire de la variable système `TimeFormat`.

Exemple 2 – fonction time()

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `transaction_time`, qui vous permettra d'afficher les résultats au format horaire sur 24 heures sans modifier la variable système `TimeFormat`.

Script de chargement

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        time(maketime(transaction_hour, transaction_minute, transaction_second),'h:mm:ss') as
transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tableau de résultats

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22
6	32	07	6:32:07
9	25	23	9:25:23
12	09	16	12:09:16
17	55	22	17:55:22
18	43	30	18:43:30
21	43	41	21:43:41

Dans cet exemple, la fonction `maketime()` est imbriquée dans la fonction `time()`. Le deuxième argument de la fonction `time()` définit le format des résultats de la fonction `maketime()` sur le format `h:mm:ss` requis.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions, chargé dans une table appelée `Transactions`.
- Heures de transaction fournies sur deux champs : `hours` et `minutes`.
- Création d'un champ, `transaction_time`, qui renvoie l'heure au format de la variable système `TimeFormat`.

Créez une mesure d'objet graphique, `transaction_time`, qui renvoie une heure au format `h:mm:ss TT`.

Script de chargement

```
SET TimeFormat='h:mm:ss TT';

Transactions:
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_amount, transaction_
quantity, customer_id
3750, 18, 43, 12423.56, 23, 2038593
3751, 6, 32, 5356.31, 6, 203521
3752, 12, 09, 15.75, 1, 5646471
3753, 21, 43, 7, 3036491
3754, 17, 55, 21484.21, 1356, 049681
3756, 2, 52, -59.18, 2, 2038593
3757, 9, 25, 3177.4, 21, 203521
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_hour`
- `transaction_minute`

Pour calculer la valeur `transaction_time`, créez la mesure suivante :

```
=maketime(transaction_hour,transaction_minute)
```

Tableau de résultats

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:00 PM
21	43	9:43:00 PM

La mesure `transaction_time` est créée dans l'objet graphique via la fonction `maketime()` et en transmettant les champs hour et minute comme arguments de la fonction.

La fonction combine ensuite ces valeurs et les secondes sont supposées être 00. Ces valeurs sont ensuite converties en un champ time, renvoyant les résultats au format de la variable système `TimeFormat`.

Exemple 4 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Créez un ensemble de données Calendar pour le mois de janvier 2022, divisé en incréments de huit heures.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpCalendar:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 31;

Left join(tmpCalendar)
load
       maketime((recno()-1)*8,00,00) as time
autogenerate 3;

Calendar:
load
       timestamp(date + time) as timestamp
resident tmpCalendar;
```

```
drop table tmpCalendar;
```

Résultats

Tableau de résultats

timestamp
1/1/2022 12:00:00 AM
1/1/2022 8:00:00 AM
1/1/2022 4:00:00 PM
1/2/2022 12:00:00 AM
1/2/2022 8:00:00 AM
1/2/2022 4:00:00 PM
1/3/2022 12:00:00 AM
1/3/2022 8:00:00 AM
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ 68 lignes supplémentaires

La fonction `autogenerate` initiale crée un calendrier contenant l'ensemble des dates de janvier dans une table appelée `tmpCalendar`.

Une deuxième table, contenant trois enregistrements, est créée. Pour chaque enregistrement, l'argument `recno() - 1` est pris (valeurs 0, 1 et 2) et le résultat est multiplié par 8. Cela génère les valeurs 0, 8 et 16. Ces valeurs sont utilisées comme paramètre horaire dans une fonction `maketime()`, avec les valeurs minute et second définies sur 0. En conséquence, la table contient trois champs horaires : 12:00:00 AM, 8:00:00 AM et 4:00:00 PM.

Cette table est jointe à la table `tmpCalendar`. Étant donné qu'il n'existe pas de champs correspondants entre les deux tables pour la jointure, les lignes des heures sont ajoutées à chaque ligne de date. C'est pourquoi chaque ligne de date est maintenant répétée trois fois avec chaque valeur horaire.

Pour finir, la table Calendar est créée à partir d'un chargement résident de la table `tmpCalendar`. Les champs date et time sont concaténés et imbriqués dans la fonction `timestamp()` pour créer le champ timestamp (horodatage).

La table `tmpCalendar` est ensuite abandonnée.

makeweekdate

Cette fonction renvoie une date calculée à partir de l'année **YYYY**, de la semaine **WW** et du jour de la semaine **D**.

Syntaxe :

```
MakeWeekDate(YYYY [ , WW [ , D ] ])
```

Type de données renvoyé : double

La fonction `makeweekdate()` est disponible comme fonction de script et de graphique. La fonction calculera la date en fonction des paramètres transmis dans la fonction. Si le paramètre day-of-week (jour de la semaine) est omis, la fonction renverra la date du lundi de cette semaine-là.

La fonction `makeweekdate()` ne prend pas en compte les variables système `BrokenWeek`, `ReferenceDay` ni `FirstWeekDay`. La semaine 1 commence le premier lundi de janvier. Par exemple, en 2022, la semaine 1 commence le 3 janvier.

Arguments

Argument	Description
YYYY	Année sous forme d'entier.
WW	Semaine sous forme d'entier. La semaine peut être positive ou négative et supérieure à 52 pour renvoyer des dates de différentes années.
D	Jour de la semaine sous forme d'entier. Si aucun jour de la semaine n'est spécifié, 0 (lundi) est utilisé. Les jours restants de la semaine sont attribués comme suit : 1 pour mardi, 2 pour mercredi, 3 pour jeudi, 4 pour vendredi, 5 pour samedi et 6 pour dimanche.

Cas d'utilisation

La fonction `makeweekdate()` est généralement utilisée dans le script pour la génération de données afin de produire une liste de dates, ou pour créer des dates lorsque l'année, la semaine et le jour de la semaine sont fournis dans les données d'entrée.

Exemples de fonction

Exemple	Résultat
`makeweekdate(2014,6,6)`	renvoie 02/09/2014
`makeweekdate(2014,6,1)`	renvoie 02/04/2014
`makeweekdate(2014,6)`	renvoie 02/03/2014 (en supposant l'utilisation du jour de la semaine 0).

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – jour inclus

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant le total des ventes hebdomadaires pour l'année 2022 dans une table appelée `Sales`.
- Dates de transaction fournies sur trois champs : `year`, `week` et `sales`.
- Chargement précédent, utilisé pour créer une mesure, `end_of_week`, via la fonction `makeweekdate()`, pour renvoyer la date du vendredi de cette semaine-là au format MM/DD/YYYY.

Pour prouver que la date renvoyée est un vendredi, l'expression `end_of_week` est également imbriquée dans la fonction `weekday()` pour afficher le jour de la semaine.

Script de chargement

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

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week,4) as end_of_week,
        weekday(makeweekdate(transaction_year, transaction_week,4)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Tableau de résultats

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Le champ `end_of_week` est créé dans l'instruction LOAD précédente à l'aide de la fonction `makeweekdate()`. Les champs `transaction_year` et `transaction_week` sont transmis via la fonction comme arguments year (année) et week (semaine). Une valeur 4 est utilisée pour l'argument day (jour).

La fonction combine ensuite ces valeurs et les convertit en un champ date, renvoyant les résultats au format de la variable système `DateFormat`.

La fonction `makeweekdate()` et ses arguments sont également imbriqués dans une fonction `weekday()` pour renvoyer le champ `week_day` ; et, comme indiqué dans le tableau ci-dessus, le champ `week_day` montre que ces dates ont lieu un vendredi.

Exemple 2 – jour exclu

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant les totaux des ventes hebdomadaires pour l'année 2022 dans une table appelée `Sales`.
- Dates de transaction fournies sur trois champs : `year`, `week` et `sales`.
- Chargement précédent utilisé pour créer une mesure, `first_day_of_week`, via la fonction `makeweekdate()`. Cela renverra la date du lundi de cette semaine-là au format MM/DD/YYYY.

Pour prouver que la date renvoyée est un lundi, l'expression `first_day_of_week` est également imbriquée dans la fonction `weekday()` pour afficher le jour de la semaine.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        day(makeweekdate(transaction_year, transaction_week)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `transaction_year`
- `transaction_week`
- `first_day_of_week`
- `week_day`

Tableau de résultats

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Mon
2022	02	01/10/2022	Mon
2022	03	01/17/2022	Mon
2022	04	01/24/2022	Mon
2022	05	01/31/2022	Mon
2022	06	02/07/2022	Mon
2022	07	02/14/2022	Mon

Le champ `first_day_of_week` est créé dans l'instruction LOAD précédente à l'aide de la fonction `makeweekdate()`. Les paramètres `transaction_year` et `transaction_week` sont transmis comme arguments de la fonction et le paramètre day (jour) est laissé vide.

La fonction combine ensuite ces valeurs et les convertit en un champ date, renvoyant les résultats au format de la variable système `DateFormat`.

La fonction `makeweekdate()` et ses arguments sont également imbriqués dans une fonction `weekday()` pour renvoyer le champ `week_day`. Comme indiqué dans le tableau ci-dessus, le champ `week_day` montre que ces dates se produisent un lundi (même si la variable `FirstWeekDay` définit le dimanche comme le premier jour de la semaine), car le paramètre day de la fonction `makeweekdate()` a été laissé vide.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant les totaux des ventes hebdomadaires pour l'année 2022 dans une table appelée `Sales`.
- Dates de transaction fournies sur trois champs : `year`, `week` et `sales`.

Dans cet exemple, un objet graphique sera utilisé pour créer une mesure équivalente au calcul `end_of_week` du premier exemple. Cette mesure utilisera la fonction `makeweekdate()` pour renvoyer la date du vendredi de cette semaine-là au format MM/DD/YYYY.

Pour prouver que la date renvoyée est un vendredi, une deuxième mesure est créée pour renvoyer le jour de la semaine.

Script de chargement

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

Master_Calendar:
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :
   - `transaction_year`
   - `transaction_week`
2. Pour effectuer le calcul équivalent à celui du champ `end_of_week` du premier exemple, créez la mesure suivante :
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Pour calculer le jour de la semaine de chaque transaction, créez la mesure suivante :
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Tableau de résultats

transaction_year	transaction_week	=makeweekdate (transaction_year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_week,4))
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Un champ équivalent au champ `end_of_week` est créé dans l'objet graphique sous forme de mesure à l'aide de la fonction `makeweekdate()`. Les champs `transaction_year` et `transaction_week` sont transmis comme arguments year (année) et week (semaine). Une valeur 4 est utilisée pour l'argument day (jour).

La fonction combine ensuite ces valeurs et les convertit en un champ date, renvoyant les résultats au format de la variable système `DateFormat`.

La fonction `makeweekdate()` et ses arguments sont également imbriqués dans une fonction `weekday()` pour renvoyer un calcul équivalent à celui du champ `week_day` du premier exemple. Comme indiqué dans le tableau ci-dessus, la dernière colonne à droite montre que ces dates se produisent un vendredi.

Exemple 4 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Dans cet exemple, créez une liste de dates contenant tous les vendredis de l'année 2022.

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Script de chargement

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

Calendar:
    load
           *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

Résultats

Tableau de résultats

date	weekday
01/07/2022	Fri
01/14/2022	Fri
01/21/2022	Fri
01/28/2022	Fri
02/04/2022	Fri
02/11/2022	Fri
02/18/2022	Fri
02/25/2022	Fri
03/04/2022	Fri
03/11/2022	Fri
03/18/2022	Fri
03/25/2022	Fri
04/01/2022	Fri
04/08/2022	Fri
04/15/2022	Fri
04/22/2022	Fri
04/29/2022	Fri
05/06/2022	Fri
05/13/2022	Fri
05/20/2022	Fri
05/27/2022	Fri
06/03/2022	Fri
06/10/2022	Fri
06/17/2022	Fri
+ 27 lignes supplémentaires	

La fonction `makeweekdate()` trouve chaque vendredi de l'année 2022. L'utilisation d'un paramètre week égal à -2 garantit qu'il ne manque aucune date. Pour finir, un chargement précédent crée un champ `weekday` supplémentaire à des fins de clarté, pour indiquer que chaque valeur `date` est un vendredi.

minute

Cette fonction renvoie un entier représentant la minute au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

Syntaxe :

```
minute(expression)
```

Type de données renvoyé : entier

Cas d'utilisation

La fonction `minute()` est utile lorsque vous souhaitez comparer des agrégations par minute. Par exemple, vous pouvez utiliser la fonction si vous souhaitez afficher la répartition du nombre d'activités par minute.

Il est possible de créer ces dimensions dans le script de chargement via la fonction permettant de créer un champ dans une table Master Calendar. Sinon, elles peuvent également être directement utilisées dans un graphique comme dimension calculée.

Exemples de fonction

Exemple	Résultat
`minute ( '09:14:36' )`	Renvoie 14.
`minute ( '0.5555' )`	Renvoie 19 (car 0.5555 = 13:19:55).

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – Variable (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des transactions par horodatage chargé dans une table appelée `Transactions`.
- La variable système `TimeStamp` par défaut (`M/D/YYYY h:mm:ss[.fff] TT`) est utilisée.
- Création d'un champ, `minute`, pour calculer la date des transactions.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        minute(timestamp) as minute
        ;
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `timestamp`
- `minute`

Tableau de résultats

timestamp	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Les valeurs du champ `minute` sont créées via la fonction `minute()` et en transmettant la valeur `timestamp` comme expression dans l'instruction LOAD précédente.

Exemple 2 – Objet graphique (graphique)

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- La variable système `TimeStamp` par défaut (`M/D/YYYY h:mm:ss[.fff] TT`) est utilisée.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Les valeurs `minute` sont calculées via une mesure dans un objet graphique.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
```

```
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `timestamp`.

Créez la mesure suivante :

```
=minute(timestamp)
```

Tableau de résultats

timestamp	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Les valeurs de `minute` sont créées via la fonction `minute()` et en transmettant la valeur `timestamp` comme expression dans une mesure de l'objet graphique.

Exemple 3 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'horodatages, généré pour représenter les entrées à une barrière de billets.
- Informations avec chaque `timestamp` et son `id` correspondant, chargées dans une table appelée `Ticket_Barrier_Tracker`.
- La variable système `TimeStamp` par défaut (`M/D/YYYY h:mm:ss[.fff] TT`) est utilisée.

L'utilisateur souhaite un objet graphique qui montre, par minute, le nombre d'entrées à la barrière.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 1;

join load
       maketime(floor(rand()*24),floor(rand()*59),floor(rand()*59)) as time
autogenerate 10000;

Ticket_Barrier_Tracker:
load
       recno() as id,
       timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez une dimension calculée à l'aide de l'expression suivante :
   `=minute(timestamp)`
3. Ajoutez la mesure d'agrégation suivante pour calculer le nombre total d'entrées :
   `=count(id)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

minute(timestamp)	=count(id)
0	174
1	171
2	175

minute(timestamp)	=count(id)
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
+ 35 lignes supplémentaires	

month

Cette fonction renvoie une valeur double : un nom de mois tel que défini dans la variable d'environnement **MonthNames** et un entier compris entre 1 et 12. Le mois est calculé à partir de l'interprétation de date de l'expression, conformément à l'interprétation standard des nombres.

La fonction renvoie le nom du mois au format de la variable système `MonthName` pour une date donnée. Elle est couramment utilisée pour créer un champ Jour comme dimension dans un calendrier principal.

Syntaxe :

```
month(expression)
```

Type de données renvoyé : entier

Exemples de fonction

Exemple	Résultat
`month( 2012-10-12 )`	renvoie Oct.
`month( 35648 )`	renvoie Aug, car 35648 = 1997-08-06.

Exemple 1 – Ensemble de données DateFormat (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système `DateFormat` est définie au format DD/MM/YYYY.
- Un chargement précédent qui crée un champ supplémentaire, appelé `month_name`, via la fonction `month()`.
- Un champ supplémentaire, appelé `long_date`, qui utilise la fonction `date()` pour exprimer la date complète.

Script de chargement

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

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        month(date) as month_name
Inline
[
date
03/01/2022
03/02/2022
03/03/2022
03/04/2022
03/05/2022
03/06/2022
03/07/2022
03/08/2022
03/09/2022
03/10/2022
```

```
03/11/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `long_date`
- `month_name`

Tableau des résultats

date	long_date	month_name
03/01/2022	03-January- 2022	Jan
03/02/2022	03-February- 2022	Feb
03/03/2022	03-March- 2022	Mar
03/04/2022	03-April- 2022	Apr
03/05/2022	03-May- 2022	May
03/06/2022	03-June- 2022	Jun
03/07/2022	03-July- 2022	Jul
03/08/2022	03-August- 2022	Aug
03/09/2022	03-September- 2022	Sep
03/10/2022	03-October- 2022	Oct
03/11/2022	03-November- 2022	Nov

Le nom du mois est correctement évalué par la fonction `month()` du script.

Exemple 2 – Dates ANSI (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système DateFormat DD/MM/YYYY est utilisée. Cependant, les dates incluses dans l'ensemble de données sont au format de date standard ANSI.

- Un chargement précédent qui crée un champ supplémentaire, appelé `month_name`, via la fonction month().
- Un champ supplémentaire, appelé `long_date`, qui utilise la fonction date() pour exprimer la date complète.

Script de chargement

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       month(date) as month_name

Inline
[
date
2022-01-11
2022-02-12
2022-03-13
2022-04-14
2022-05-15
2022-06-16
2022-07-17
2022-08-18
2022-09-19
2022-10-20
2022-11-21
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `long_date`
- `month_name`

Tableau des résultats

date	long_date	month_name
03/11/2022	11-March- 2022	11
03/12/2022	11-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16

date	long_date	month_name
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Le nom du mois est correctement évalué par la fonction `month()` du script.

Exemple 3 – Dates non formatées (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de dates appelé `Master_Calendar`. La variable système `DateFormat` DD/MM/YYYY est utilisée.
- Un chargement précédent qui crée un champ supplémentaire, appelé `month_name`, via la fonction `month()` .
- La date non formatée d'origine, appelée `unformatted_date`.
- Un champ supplémentaire, appelé `long_date`, qui utilise la fonction `date()` pour exprimer la date complète.

Script de chargement

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

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      month(unformatted_date) as month_name

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
```

```
45048
45078
45008
45038
45068
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `unformatted_date`
- `long_date`
- `month_name`

Tableau des résultats

unformatted_date	long_date	month_name
44868	03-January- 2022	Jan
44898	03-February- 2022	Feb
44928	03-March- 2022	Mar
44958	03-April- 2022	Apr
44988	03-May- 2022	May
45018	03-June- 2022	Jun
45048	03-July- 2022	Jul
45078	03-August- 2022	Aug
45008	03-September- 2022	Sep
45038	03-October- 2022	Oct
45068	03-November- 2022	Nov

Le nom du mois est correctement évalué par la fonction `month()` du script.

Exemple 4 – Calcul du mois d'expiration

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données de commandes passées en mars appelé `Subscriptions`. La table contient trois champs :
    - id
    - order_date
    - amount

Script de chargement

```
Subscriptions:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `order_date`.

Pour calculer le mois d'expiration d'une commande, créez la mesure suivante : `=month(order_date+180)`.

Tableau des résultats

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Aug
03/03/2022	Aug
03/04/2022	Sep
03/05/2022	Oct
03/06/2022	Nov

order_date	=month(order_date+180)
03/07/2022	Dec
03/08/2022	Jan
03/09/2022	Mar
03/10/2022	Apr
03/11/2022	May

La fonction `month()` détermine correctement qu'une commande passée le 11 mars expirera en juillet.

monthend

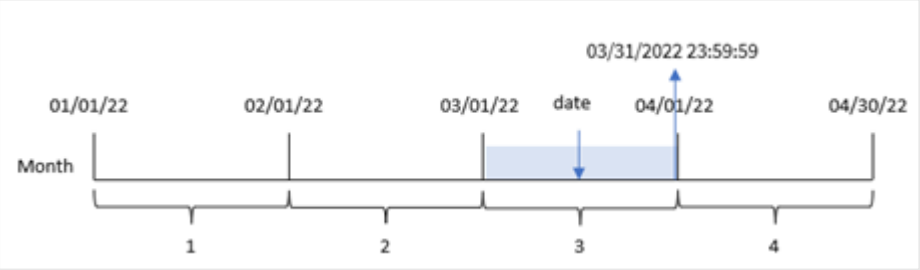
Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour du mois contenant l'argument `date`. Le format de sortie par défaut correspond à l'argument `DateFormat` défini dans le script.

Syntaxe :

```
MonthEnd(date[, period_no])
```

En d'autres termes, la fonction `monthend()` détermine le mois au cours duquel tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la dernière milliseconde de ce mois-là.

Diagramme de la fonction `monthend`.



Cas d'utilisation

La fonction `monthend()` est utilisée dans le cadre d'une expression lorsque vous souhaitez que le calcul utilise la fraction du mois qui n'a pas encore eu lieu. Par exemple, si vous souhaitez calculer le total des intérêts non encore encourus au cours du mois.

Type de données renvoyé : double

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier qui, s'il est égal à 0 ou s'il est omis, indique le mois contenant la **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les mois passés tandis que les valeurs positives désignent les mois à venir.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`monthend('02/19/2012')`	Renvoie 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Renvoie 01/31/2001 23:59:59.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée '`Transactions`'.
- Champ date dans la variable système `DateFormat` au format (`MM/DD/YYYY`).
- Instruction LOAD précédente contenant les éléments suivants :
  - La fonction `monthend()` définie comme le champ '`end_of_month`'.
  - La fonction `timestamp` définie comme le champ '`end_of_month_timestamp`'.

Script de chargement

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

Transactions:
    Load
    *,
    monthend(date) as end_of_month,
```

```
    timestamp(monthend(date)) as end_of_month_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- end_of_month
- end_of_month_timestamp

Tableau de résultats

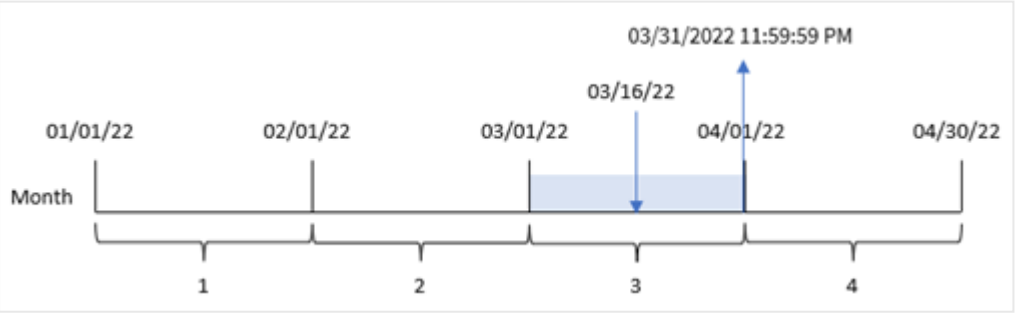
id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM

id	date	end_of_month	end_of_month_timestamp
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Le champ '`end_of_month`' est créé dans l'instruction LOAD précédente via la fonction `monthend()` et en transmettant le champ date comme argument de la fonction.

La fonction `monthend()` identifie le mois dans lequel tombe la valeur date en renvoyant un horodatage pour la dernière milliseconde de ce mois.

Diagramme de la fonction `monthend` avec March (mars) comme le mois sélectionné.



La transaction 8192 a eu lieu le 16 mars. La fonction `monthend()` renvoie la dernière milliseconde de ce mois, soit le 31 mars à 11:59:59 PM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, la tâche consiste à créer un champ, ‘previous_month_end’, qui renvoie l'horodatage de la fin du mois avant la réalisation de la transaction.

Script de chargement

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

Transactions:
    Load
    *,
    monthend(date,-1) as previous_month_end,
    timestamp(monthend(date,-1)) as previous_month_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

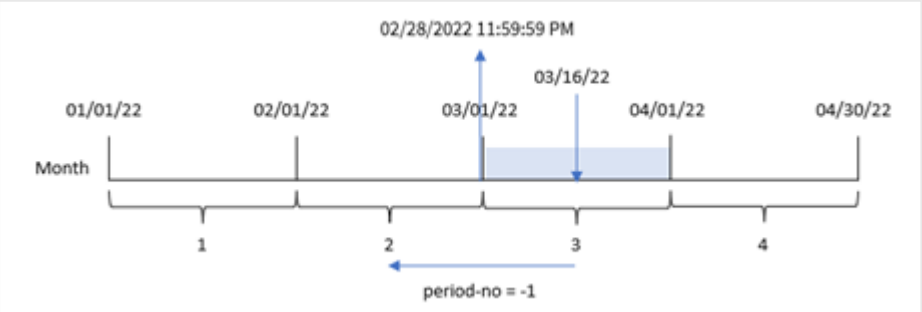
- id
- date
- previous_month_end
- previous_month_end_timestamp

Tableau de résultats

id	date	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

La fonction `monthend()` commence par identifier le mois au cours duquel les transactions ont lieu, avec un argument `period_no` défini sur `-1` utilisé comme argument de décalage. Elle décale ensuite d'un mois en arrière et identifie la dernière milliseconde de ce mois-là.

Diagramme de la fonction `monthend` avec la variable period_no.



La transaction 8192 a eu lieu le 16 mars. La fonction `monthend()` identifie que le mois avant la transaction était le mois de février. Elle renvoie alors la dernière milliseconde de ce mois, le 28 février à 11:59:59 PM.

Exemple 3 - exemple graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, l'ensemble de données est inchangé et chargé dans l'application. La tâche consiste à créer un calcul qui renvoie un horodatage pour la fin du mois de réalisation des transactions sous forme de mesure dans un graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `id`

Pour calculer la date de fin du mois d'une transaction, créez les mesures suivantes :

- `=monthend(date)`
- `=timestamp(monthend(date))`

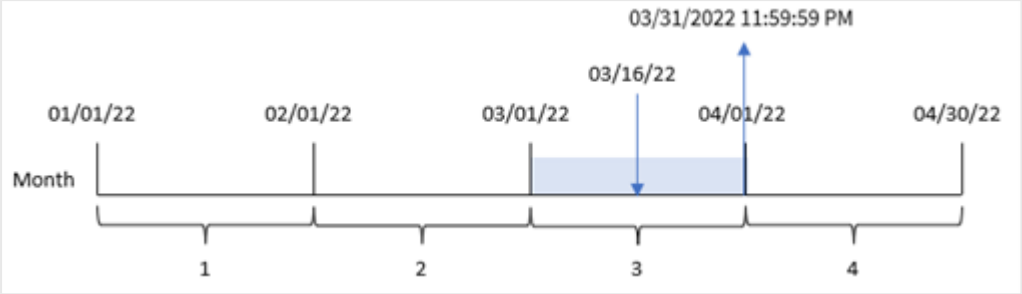
Tableau de résultats

id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	1/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	1/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8205	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8206	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8207	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM

La mesure '`end_of_month`' est créée dans le graphique via la fonction `monthend()` et en transmettant le champ date comme argument de la fonction.

La fonction `monthend()` identifie le mois dans lequel tombe la valeur date et renvoie un horodatage pour la dernière milliseconde de ce mois.

Diagramme de la fonction `monthend` avec la variable period_no.



La transaction 8192 a eu lieu le 16 mars. La fonction `monthend()` renvoie la dernière milliseconde de ce mois, soit le 31 mars à 11:59:59 PM.

Exemple 4 – scénario

Script de chargement et résultats

Vue d'ensemble

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Employee_Expenses`'. La table contient les champs suivants :

- ID des employés
- Nom des employés
- Notes de frais quotidiennes moyennes de chaque employé

L'utilisateur final souhaite un graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées pour le reste du mois.

Script de chargement

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- employee_id
- employee_name

Pour calculer les intérêts accumulés, créez la mesure suivante :

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

Cette mesure est dynamique est produira des résultats de table différents suivant la date de chargement des données.

Définissez le **Formatage des nombres** de la mesure sur **Money** (Devise).

Tableau de résultats

employee_id	employee_name	=floor(monthend(today(1),0)-today(1))*avg_daily_claim
182	Mark	$30.00
183	Deryck	$25.00
184	Dexter	$25.00
185	Sydney	$54.00
186	Agatha	$36.00

La fonction `monthend()` renvoie la date de fin du mois en cours en utilisant la date d'aujourd'hui comme seul argument. L'expression renvoie le nombre de jours restants pour ce mois en soustrayant la date d'aujourd'hui de la date de fin du mois.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes par employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours du mois restant.

monthname

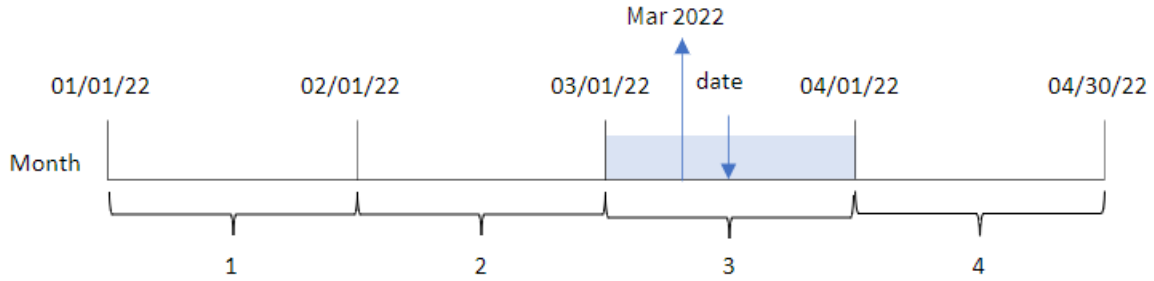
Cette fonction renvoie une valeur d'affichage présentant le mois (formaté selon la variable de script **MonthNames**) et l'année avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour du mois.

Syntaxe :

```
MonthName(date[, period_no])
```

Type de données renvoyé : double

Diagramme de la fonction monthname



Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier qui, s'il est égal à 0 ou s'il est omis, indique le mois contenant la **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les mois passés tandis que les valeurs positives désignent les mois à venir.

Exemples de fonction

Exemple	Résultat
`monthname('10/19/2013')`	Renvoie Oct 2013.
`monthname('10/19/2013', -1)`	Renvoie Sep 2013.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `transaction_month`, qui renvoie le mois au cours duquel les transactions ont eu lieu.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date) as transaction_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

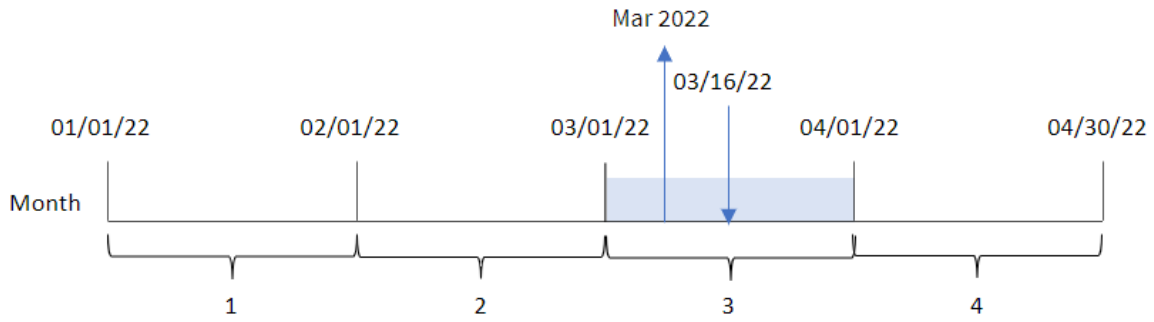
- `date`
- `transaction_month`

Tableau de résultats

date	transaction_month
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Le champ `transaction_month` est créé dans l'instruction LOAD précédente via la fonction `monthname()` et en transmettant le champ `date` comme argument de la fonction.

Diagramme de la fonction monthname, exemple de base



La fonction `monthname()` identifie que la transaction 8192 a eu lieu en mars 2022 et renvoie cette valeur à l'aide de la variable système `MonthNames`.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario inline que ceux du premier exemple.
- Création d'un champ, `transaction_previous_month`, qui renvoie l'horodatage de la fin du mois avant la réalisation des transactions.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date,-1) as transaction_previous_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `transaction_previous_month`

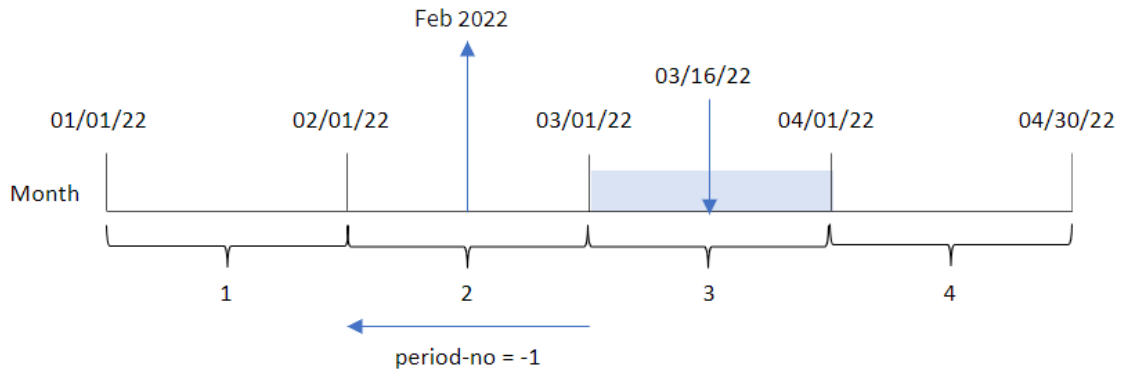
Tableau de résultats

date	transaction_previous_month
1/7/2022	Dec 2021
1/19/2022	Dec 2021
2/5/2022	Jan 2022
2/28/2022	Jan 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Apr 2022
5/16/2022	Apr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022
8/2/2022	Jul 2022
8/8/2022	Jul 2022

date	transaction_previous_month
8/19/2022	Jul 2022
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

Dans cet exemple, étant donné que la valeur `period_no` de -1 a été utilisée comme argument de décalage dans la fonction `monthname()`, la fonction commence par identifier le mois au cours duquel les transactions ont lieu. Elle passe ensuite au mois précédent et renvoie le nom du mois et l'année.

Diagramme de la fonction monthname, exemple period_no



La transaction 8192 a eu lieu le 16 mars. La fonction `monthname()` identifie que le mois précédant la transaction était février et renvoie le mois, au format de la variable système `MonthNames`, ainsi que l'année 2022.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario inline que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant à la fin du mois de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
```

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.

Créez la mesure suivante :

```
=monthname(date)
```

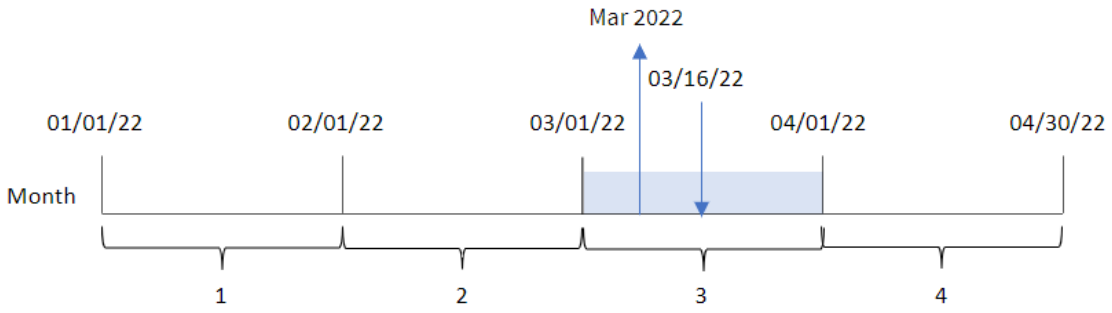
Tableau des résultats

date	=monthname(date)
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022

date	=monthname(date)
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

La mesure `month_name` est créée dans l'objet graphique via la fonction `monthname()` et en transmettant le champ `date` comme argument de la fonction.

Diagramme de la fonction monthname, exemple objet graphique



La fonction `monthname()` identifie que la transaction 8192 a eu lieu en mars 2022 et renvoie cette valeur à l'aide de la variable système `MonthNames`.

monthsend

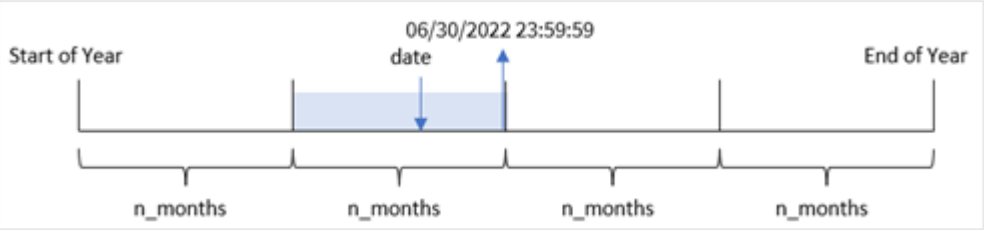
Cette fonction renvoie une valeur correspondant à l'horodatage de la dernière milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence. Il est également possible de rechercher l'horodatage pour la fin d'une période passée ou future. Le format de sortie par défaut correspond au format de date `DateFormat` défini dans le script.

Syntaxe :

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction `monthsend`.



Arguments

Argument	Description
n_months	Nombre de mois définissant la période. Entier ou expression aboutissant à un entier qui doit correspondre à l'une de ces valeurs : 1 (qui équivaut à la fonction inmonth()), 2 (période de deux mois), 3 (qui équivaut à la fonction inquarter()), 4 (période de quatre mois) ou 6 (semestre).
date	Date ou horodatage à évaluer.
period_no	Il est possible de décaler la période à l'aide de l'argument **period_no**, d'un entier ou d'une expression aboutissant à un entier, où la valeur 0 indique la période comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les périodes passées tandis que les valeurs positives désignent les périodes à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

La fonction `monthsend()` divise l'année en segments en fonction de l'argument `n_months` fourni. Elle évalue ensuite le segment dans lequel tombe chaque date fournie et renvoie la dernière milliseconde, au format date, de ce segment. La fonction peut renvoyer l'horodatage de fin de segments précédents ou suivants ainsi que redéfinir le premier mois de l'année.

Les segments suivants de l'année sont disponibles dans la fonction en tant qu'arguments `n_month`.

Arguments `n_month`

Période	Nombre de mois
mois	1
période de deux mois	2
trimestre	3
période de quatre mois	4
semestre	6

Cas d'utilisation

La fonction `monthsend()` est utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction du mois qui s'est écoulée jusqu'à présent. En utilisant une variable, l'utilisateur a la possibilité de sélectionner la période de son choix. Par exemple, la fonction `monthsend()` peut fournir une variable d'entrée permettant à l'utilisateur de calculer le total des intérêts non encore encourus au cours du mois, du trimestre ou du semestre.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`monthsend(4, '07/19/2013')`	Renvoie 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Renvoie 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Renvoie 01/31/2014. Car le début de l'année devient le mois numéro 2.

Exemple 1 - exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Un ensemble de données contenant un ensemble de transactions pour 2022 est chargé dans une table appelée '`Transactions`'.
- Champ date fourni dans la variable système `DateFormat` au format `(MM/DD/YYYY)`.

- Instruction LOAD précédente contenant les éléments suivants :
    - La fonction `monthsend` définie comme le champ '`bi_monthly_end`'. Cela regroupe les transactions en segments bimestriels.
    - Fonction `timestamp` qui renvoie l'horodatage de début du segment de chaque transaction.

Script de chargement

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

Transactions:
    Load
    *,
    monthsend(2,date) as bi_monthly_end,
    timestamp(monthsend(2,date)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

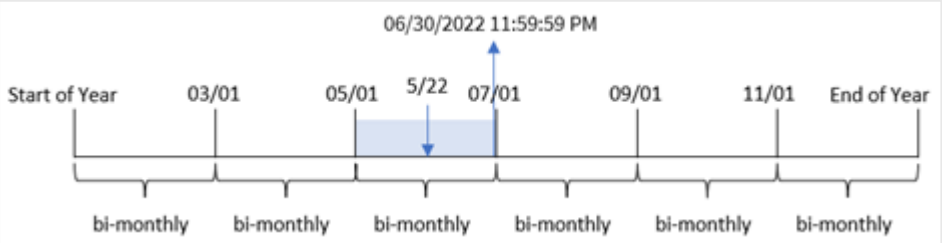
- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

Tableau de résultats

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	1/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	1/31/2022 11:59:59 PM

Le champ '`bi_monthly_end`' est créé dans l'instruction LOAD précédente à l'aide de la fonction `monthsend()`. Le premier argument fourni est `2`, divisant l'année en segments bimestriels. Le deuxième argument identifie le champ en cours d'évaluation.

Diagramme de la fonction `monthsend` avec des segments bimestriels.



La transaction 8195 a lieu le 22 mai. La fonction `monthsend()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. En conséquence, la fonction renvoie la dernière milliseconde de ce segment, 06/30/2022 11:59:59 PM.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, la tâche consiste à créer un champ, '`prev_bi_monthly_end`', qui renvoie la première millisecondes du segment bimestriel avant la réalisation de la transaction.

Script de chargement

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

Transactions:
    Load
    *,
    monthsend(2,date,-1) as prev_bi_monthly_end,
    timestamp(monthsend(2,date,-1)) as prev_bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

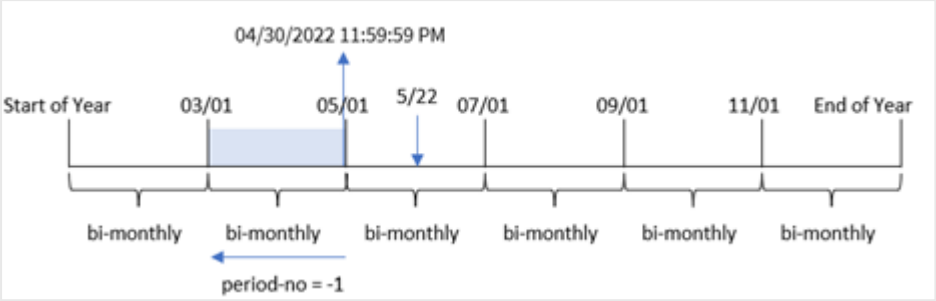
- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

Tableau de résultats

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/22/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

Si on utilise `-1` comme argument `period_no` dans la fonction `monthsend()`, après avoir initialement divisé une année en segments bimestriels, la fonction renvoie la dernière milliseconde du segment bimestriel précédent avant la transaction.

Diagramme de la fonction `monthsend` renvoyant le précédent segment bimestriel.



La transaction 8195 a lieu dans le segment entre mai et juin. En conséquence, le segment bimestriel précédent était compris entre le 1er mars et le 30 avril et la fonction renvoie donc la dernière milliseconde de ce segment, 04/30/2022 11:59:59 PM.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Dans cet exemple, la stratégie organisationnelle exige qu'avril soit le premier mois de l'exercice financier.

Créez un champ, '`bi_monthly_end`', qui regroupe les transactions en segments bimestriels et renvoie l'horodatage de la dernière milliseconde du segment pour chaque transaction.

Script de chargement

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

Transactions:
    Load
    *,
    monthsend(2,date,0,4) as bi_monthly_end,
    timestamp(monthsend(2,date,0,4)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

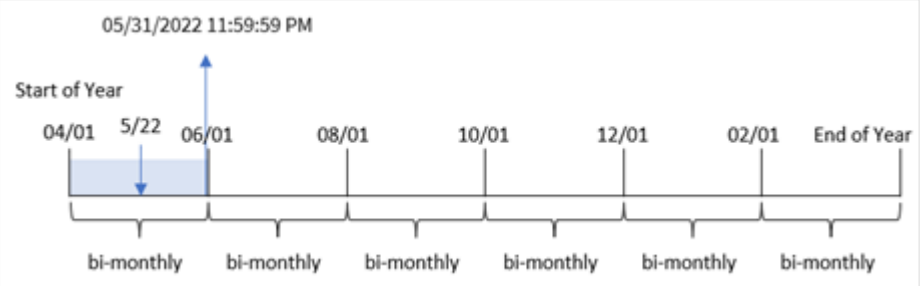
Tableau de résultats

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/22/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	6/26/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	bi_monthly_end	bi_monthly_end_timestamp
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Si on utilise `4` comme argument `first_month_of_year` dans la fonction `monthsend()`, la fonction commence l'année le 1er avril. Elle divise ensuite l'année en segments bimestriels : Apr-May, Jun-Jul, Aug-Sep, Oct-Nov, Dec-Jan, Feb-Mar.

Diagramme de la fonction `monthsend` avec le premier mois de l'année défini sur avril



La transaction 8195 a eu lieu le 22 mai et tombe dans le segment entre le 1er avril et le 31 mai. En conséquence, la fonction renvoie la dernière milliseconde de ce segment, 05/31/2022 11:59:59 PM.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés. Cependant, dans cet exemple, l'ensemble de données est inchangé et chargé dans l'application.

Dans cet exemple, la tâche consiste à créer un calcul qui regroupe les transactions en segments bimestriels et renvoie l'horodatage de la dernière milliseconde du segment pour chaque transaction sous forme de mesure dans un objet graphique d'une application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
```

```
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

```
date
```

Pour récupérer l'horodatage de la dernière milliseconde du segment bimestriel de la transaction, créez les mesures suivantes :

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

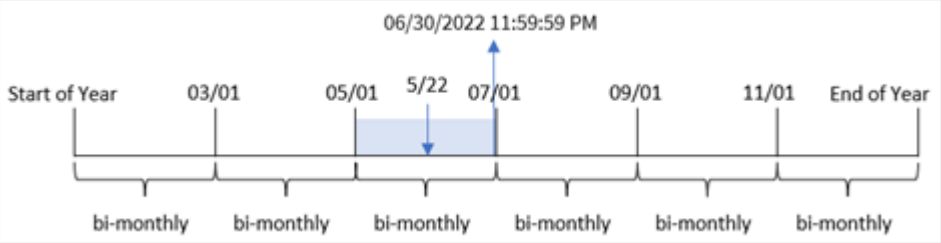
Tableau de résultats

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	1/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	1/31/2022 11:59:59 PM

Le champ '`bi_monthly_end`' est créé sous forme de mesure dans l'objet graphique à l'aide de la fonction `monthsend()`. Le premier argument fourni est `2`, qui divise l'année en segments bimestriels. Le deuxième argument identifie le champ en cours d'évaluation.

Diagramme de la fonction `monthsend` avec des segments bimestriels.



La transaction 8195 a lieu le 22 mai. La fonction `monthsend()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. En conséquence, la fonction renvoie la première milliseconde de ce segment, 06/30/2022 11:59:59 PM.

Exemple 5 – scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Dans cet exemple, un ensemble de données est chargé dans une table nommée '`Employee_Expenses`'. La table contient les champs suivants :

- ID des employés
- Nom des employés
- Notes de frais quotidiennes moyennes de chaque employé

L'utilisateur final souhaite un graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées pour le reste d'une période de son choix. L'exercice financier commence en janvier.

Script de chargement

```
SET vPeriod = 1;

Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Chargez les données et ouvrez une nouvelle feuille.

Au début du script de chargement, une variable, `vPeriod`, est créée ; elle est liée au contrôle d'entrée de variable.

Procédez comme suit :

1. Dans le panneau des ressources, cliquez sur **Objets personnalisés**.
2. Sélectionnez **Qlik Dashboard bundle** et créez un objet **Entrée de variable**.
3. Saisissez un titre pour l'objet graphique.
4. Sous **Variable**, sélectionnez **vPeriod** comme nom et définissez l'objet de sorte qu'il s'affiche sous forme de **Liste déroulante**.
5. Sous **Valeurs**, cliquez sur les valeurs **Dynamiques**. Saisissez les éléments suivants : `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Créez une table et ces champs comme dimensions :

- `employee_id`
- `employee_name`

Pour calculer les intérêts accumulés, créez la mesure suivante :

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Cette mesure est dynamique est produira des résultats de table différents suivant la date de chargement des données.

Définissez le **Formatage des nombres** de la mesure sur **Money** (Devise).

Tableau de résultats

employee_id	employee_ name	=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_ claim
182	Mark	$1410.00
183	Deryck	$1175.00
184	Dexter	$1175.00
185	Sydney	$2538.00
186	Agatha	$1692.00

La fonction `monthsend()` utilise l'entrée de l'utilisateur comme premier argument et la date d'aujourd'hui comme deuxième argument. Cela renvoie la date de fin de la période sélectionnée par l'utilisateur. L'expression renvoie ensuite le nombre de jours restants pour la période sélectionnée en soustrayant la date d'aujourd'hui de cette date de fin.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes par employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours des jours restants de cette période.

monthsname

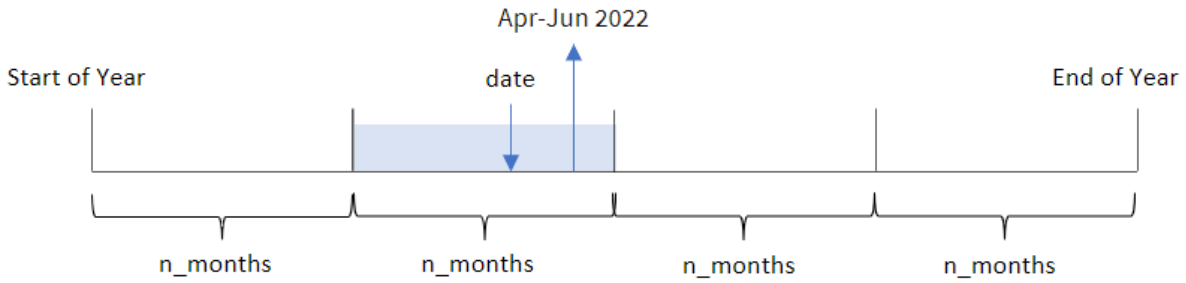
Cette fonction renvoie une valeur d'affichage représentant la plage des mois de la période (formatée d'après la variable de script **MonthNames**) de même que l'année. La valeur numérique sous-jacente correspond à un horodatage de la première milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence.

Syntaxe :

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction monthsname



La fonction `monthsname()` divise l'année en segments en fonction de l'argument `n_months` fourni. Elle évalue ensuite le segment auquel appartient chaque valeur `date` fournie et renvoie les noms de mois de début et de fin de ce segment, ainsi que l'année. La fonction permet également de renvoyer ces limites à partir des segments précédents ou suivants, ainsi que de redéfinir le premier mois de l'année.

Les segments suivants de l'année sont disponibles dans la fonction en tant qu'arguments `n_month` :

Arguments `n_month` possibles

Périodes	Nombre de mois
mois	1
période de deux mois	2
trimestre	3
période de quatre mois	4
semestre	6

Arguments

Argument	Description
n_months	Nombre de mois définissant la période. Entier ou expression aboutissant à un entier qui doit correspondre à l'une de ces valeurs : 1 (qui équivaut à la fonction inmonth()), 2 (période de deux mois), 3 (qui équivaut à la fonction inquarter()), 4 (période de quatre mois) ou 6 (semestre).
date	Date ou horodatage à évaluer.
period_no	Il est possible de décaler la période à l'aide de l'argument **period_no**, d'un entier ou d'une expression aboutissant à un entier, où la valeur 0 indique la période comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les périodes passées tandis que les valeurs positives désignent les périodes à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `monthsname()` est utile lorsque vous souhaitez permettre à l'utilisateur de comparer des agrégations en fonction d'une période de son choix. Par exemple, vous pouvez fournir une variable d'entrée pour permettre à l'utilisateur de voir les ventes totales de produits par mois, trimestre ou semestre.

Ces dimensions peuvent être créées soit dans le script de chargement, en ajoutant la fonction sous forme de champ dans une table Master Calendar, soit en créant la dimension directement dans un graphique sous forme de dimension calculée.

Exemples de fonction

Exemple	Résultat
`monthsname(4, '10/19/2013')`	Renvoie 'Sep-Dec 2013'. Dans cet exemple et dans les autres, l'instruction **SET Monthnames** est définie sur Jan;Feb;Mar, etc.
`monthsname(4, '10/19/2013', -1)`	Renvoie 'May-Aug 2013'.
`monthsname(4, '10/19/2013', 0, 2)`	Renvoie 'Oct-Jan 2014', car l'année est indiquée comme commençant le mois 2. Par conséquent, la période de quatre mois se termine le premier mois de l'année suivante.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `bi_monthly_range`, qui regroupe les transactions en segments bimestriels et renvoie les noms de limite de ce segment pour chaque transaction.

Script de chargement

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

Transactions:
    Load
        *,
        monthsname(2,date) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

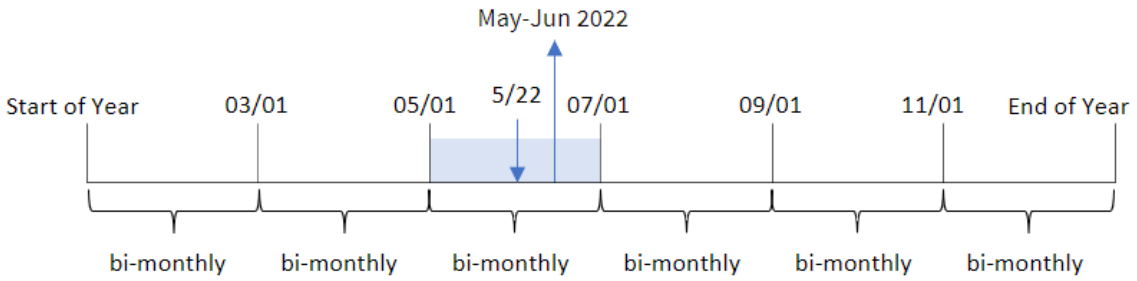
- `date`
- `bi_monthly_range`

Tableau de résultats

date	bi_monthly_range
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Le champ `bi_monthly_range` est créé dans l'instruction LOAD précédente à l'aide de la fonction `monthsname()`. Le premier argument fourni est 2, divisant l'année en segments bimestriels. Le deuxième argument identifie le champ en cours d'évaluation.

Diagramme de la fonction monthsname, exemple de base



La transaction 8195 a lieu le 22 mai. La fonction `monthsname()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. Par conséquent, la fonction renvoie ces mois au format de la variable système `MonthNames`, ainsi que l'année, mai-juin 2022.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario inline que ceux du premier exemple.
- Création d'un champ, `prev_bi_monthly_range`, qui regroupe les transactions en segments bimestriels et renvoie les noms de limite de segment précédent pour chaque transaction.

Ajoutez un autre texte ici, si nécessaire, avec des listes, etc.

Script de chargement

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

Transactions:
    Load
        *,
        MonthsName(2,date,-1) as prev_bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
```

```
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `prev_bi_monthly_range`

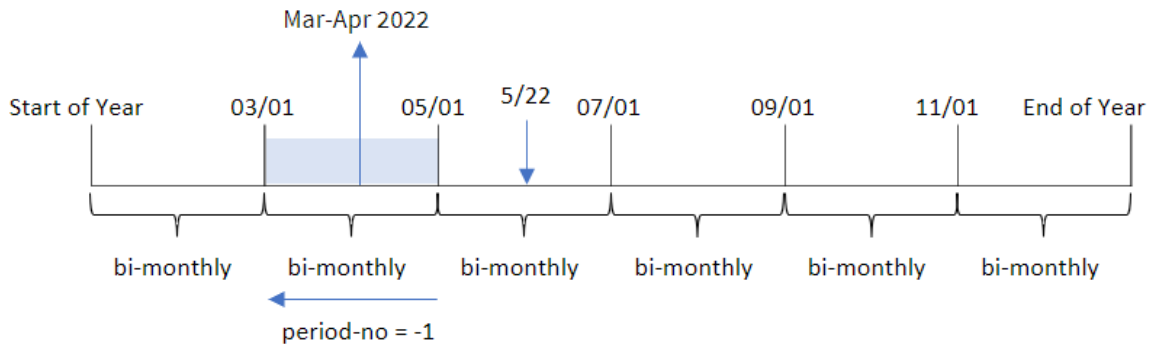
Tableau de résultats

date	prev_bi_monthly_range
2/19/2022	Nov-Dec 2021
3/7/2022	Jan-Feb 2022
3/30/2022	Jan-Feb 2022
4/5/2022	Jan-Feb 2022
4/16/2022	Jan-Feb 2022
5/1/2022	Mar-Apr 2022
5/7/2022	Mar-Apr 2022
5/22/2022	Mar-Apr 2022
6/15/2022	Mar-Apr 2022
6/26/2022	Mar-Apr 2022
7/9/2022	May-Jun 2022
7/22/2022	May-Jun 2022
7/23/2022	May-Jun 2022
7/27/2022	May-Jun 2022

date	prev_bi_monthly_range
8/2/2022	May-Jun 2022
8/8/2022	May-Jun 2022
8/19/2022	May-Jun 2022
9/26/2022	Jul-Aug 2022
10/14/2022	Jul-Aug 2022
10/29/2022	Jul-Aug 2022

Dans cet exemple, -1 est utilisé comme argument `period_no` dans la fonction `monthsname()`. Après avoir initialement divisé une année en segments bimestriels, la fonction renvoie les limites de segment précédentes correspondant au moment où une transaction a lieu.

Diagramme de la fonction monthsname, exemple period_no



La transaction 8195 a lieu dans le segment entre mai et juin. Par conséquent, le segment bimestriel précédent se situait entre le 1er mars et le 30 avril. La fonction renvoie donc Mar-Apr 2022.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario inline que ceux du premier exemple.
- Création d'un champ différent, `bi_monthly_range`, qui regroupe les transactions en segments bimestriels et renvoie les limites de segment de chaque transaction.

Cependant, dans cet exemple, nous devons également définir avril comme le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
        *,
        MonthsName(2,date,0,4) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `bi_monthly_range`

Tableau de résultats

date	bi_monthly_range
2/19/2022	Feb-Mar 2021
3/7/2022	Feb-Mar 2021

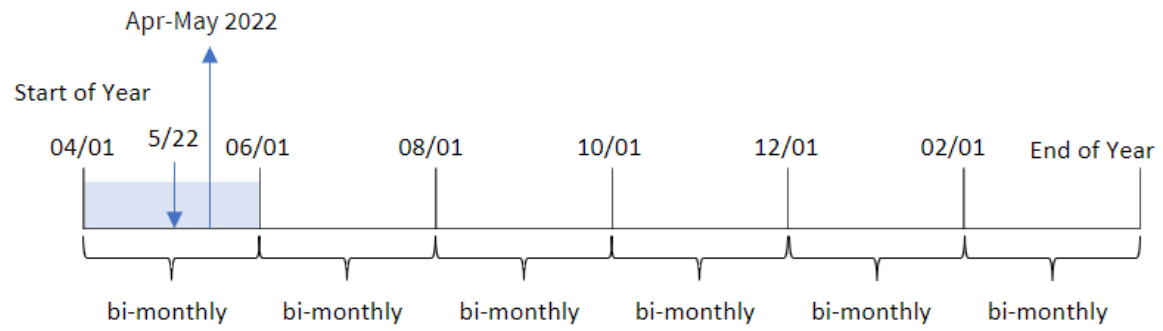
date	bi_monthly_range
3/30/2022	Feb-Mar 2021
4/5/2022	Apr-May 2022
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Si on utilise 4 comme argument `first_month_of_year` dans la fonction `monthsname()`, la fonction commence l'année le 1er avril, puis divise l'année en segments bimestriels : Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Texte de paragraphe des résultats.

La transaction 8195 a eu lieu le 22 mai et tombe dans le segment entre le 1er avril et le 31 mai. Par conséquent, la fonction renvoie Apr-May 2022.

Diagramme de la fonction monthsname, exemple first_month_of_year



Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario inline que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui regroupe les transactions en segments bimestriels et renvoie les limites de segment de chaque transaction est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
```

```
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.

Créez la mesure suivante :

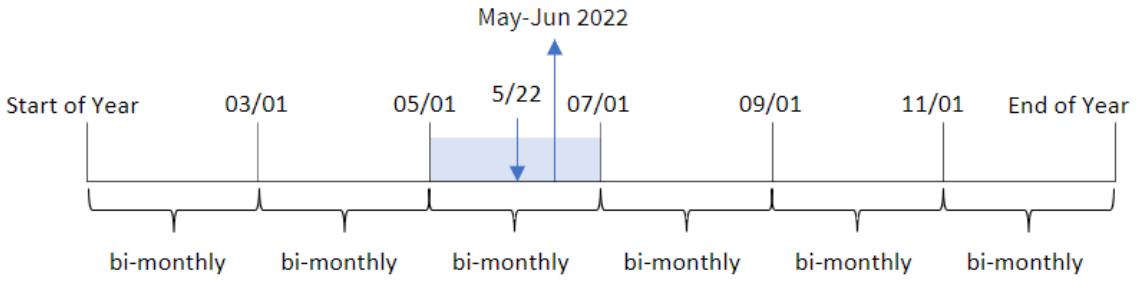
```
=monthsname(2,date)
```

Tableau des résultats

date	=monthsname(2,date)
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Le champ `bi_monthly_range` est créé sous forme de mesure dans l'objet graphique à l'aide de la fonction `monthsname()`. Le premier argument fourni est 2, divisant l'année en segments bimestriels. Le deuxième argument identifie le champ en cours d'évaluation.

Diagramme de la fonction monthsname, exemple objet graphique



La transaction 8195 a lieu le 22 mai. La fonction `monthsname()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. Par conséquent, la fonction renvoie ces mois au format de la variable système `MonthNames`, ainsi que l'année, May-Jun 2022.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

L'utilisateur final souhaite un objet graphique qui affiche les ventes totales en fonction d'une période de son choix. Cela est possible même lorsque cette dimension n'est pas disponible dans le modèle de données, en utilisant la fonction `monthsname()` comme dimension calculée modifiée dynamiquement par un contrôle d'entrée de variable.

Script de chargement

```
SET vPeriod = 1;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
```

```
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille.

Au début du script de chargement, une variable (`vPeriod`) a été créée ; elle sera liée au contrôle d'entrée de variable. Ensuite, configurez la variable comme un objet personnalisé dans la feuille.

Procédez comme suit :

1. Dans le panneau des ressources, cliquez sur **Objets personnalisés**.
2. Sélectionnez **Qlik Dashboard bundle** et créez un objet **Entrée de variable**.
3. Saisissez un titre pour l'objet graphique.
4. Sous **Variable**, sélectionnez **vPeriod** comme nom et définissez l'objet de sorte qu'il s'affiche sous forme de **Liste déroulante**.
5. Sous **Valeurs**, configurez l'objet de sorte qu'il utilise des valeurs dynamiques. Saisissez les éléments suivants :
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Ensuite, créez le tableau de résultats.

Procédez comme suit :

1. Créez un tableau et ajoutez la dimension calculée suivante :
   `=monthsname($(vPeriod),date)`
2. Ajoutez cette mesure pour calculer les ventes totales :
   `=sum(amount)`

3. Définissez le **Formatage des nombres** des mesures sur **Money** (Devise). Cliquez sur ✔ **Édition terminée**. Vous pouvez maintenant modifier les données affichées dans le tableau en ajustant le segment de temps dans l'objet variable.

Voici ce à quoi ressemblera le tableau de résultats lorsque l'option `tertial` est sélectionnée :

Tableau de résultats

monthsname($(vPeriod),date)	=sum(amount)
Jan-Apr 2022	253.89
Mai-Août 2022	713.58
Sep-Dec 2022	248.12

monthsstart

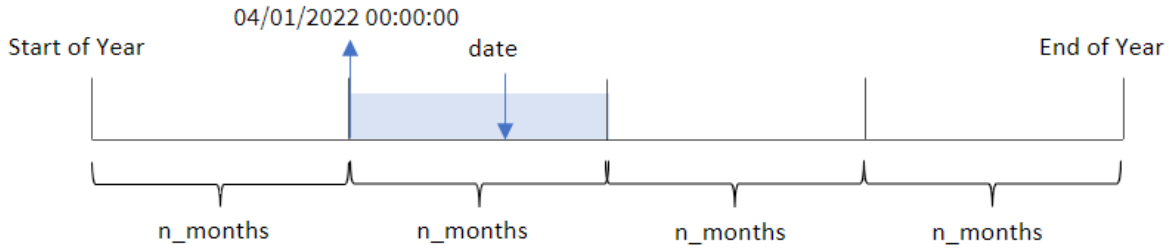
Cette fonction renvoie une valeur correspondant à l'horodatage de la première milliseconde du mois, de la période de deux mois, du trimestre, de la période de quatre mois ou du semestre contenant une date de référence. Il est également possible de rechercher l'horodatage d'une période passée ou future.Le format de sortie par défaut correspond au format de date **DateFormat** défini dans le script.

Syntaxe :

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction `monthsstart()`



La fonction `monthsstart()` divise l'année en segments en fonction de l'argument `n_months` fourni. Elle évalue ensuite le segment dans lequel tombe chaque date fournie et renvoie la première milliseconde de ce segment, au format date. La fonction permet également de renvoyer l'horodatage de début à partir des segments précédents ou suivants, ainsi que de redéfinir le premier mois de l'année.

Les segments suivants de l'année sont disponibles dans la fonction en tant qu'arguments `n_month` :

Arguments `n_month` possibles

Périodes	Nombre de mois
mois	1
période de deux mois	2
trimestre	3
période de quatre mois	4
semestre	6

Arguments

Argument	Description
n_months	Nombre de mois définissant la période. Entier ou expression aboutissant à un entier qui doit correspondre à l'une de ces valeurs : 1 (qui équivaut à la fonction inmonth()), 2 (période de deux mois), 3 (qui équivaut à la fonction inquarter()), 4 (période de quatre mois) ou 6 (semestre).
date	Date ou horodatage à évaluer.
period_no	Il est possible de décaler la période à l'aide de l'argument **period_no**, d'un entier ou d'une expression aboutissant à un entier, où la valeur 0 indique la période comprenant l'argument **base_date**. Les valeurs négatives spécifiées pour **period_no** indiquent les périodes passées tandis que les valeurs positives désignent les périodes à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `monthsstart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction d'une période qui n'a pas encore eu lieu. Cela peut être utilisé, par exemple, pour fournir une variable d'entrée permettant à l'utilisateur de calculer le total des intérêts accumulés jusqu'ici au cours du mois, du trimestre ou du semestre.

Exemples de fonction

Exemple	Résultat
`monthsstart(4, '10/19/2013')`	Renvoie 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Renvoie 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2)`	Renvoie 10/01/2013, car le début de l'année devient le mois 2.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `bi_monthly_start`, qui regroupe les transactions en segments bimestriels et renvoie l'horodatage de début du segment pour chaque transaction.

Script de chargement

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

Transactions:
    Load
        *,
        monthsstart(2,date) as bi_monthly_start,
        timestamp(monthsstart(2,date)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
```

```
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

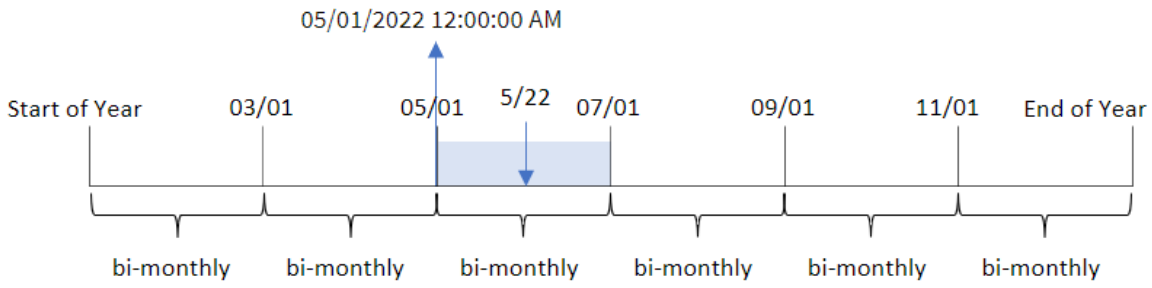
Tableau de résultats

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	01/01/2022	1/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Le champ `bi_monthly_start` est créé dans l'instruction LOAD précédente à l'aide de la fonction `monthsstart()`. Le premier argument fourni est 2, divisant l'année en segments bimestriels. Le deuxième argument identifie le champ en cours d'évaluation.

Diagramme de la fonction `monthsstart()`, exemple sans argument supplémentaire



La transaction 8195 a lieu le 22 mai. La fonction `monthsstart()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. Par conséquent, la fonction renvoie la première milliseconde de ce segment, à savoir, le 1er mai 2022 à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `prev_bi_monthly_start`, qui renvoie la première millisecondes du segment bimestriel avant la réalisation de la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        monthsstart(2,date,-1) as prev_bi_monthly_start,
           timestamp(monthsstart(2,date,-1)) as prev_bi_monthly_start_timestamp
           ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

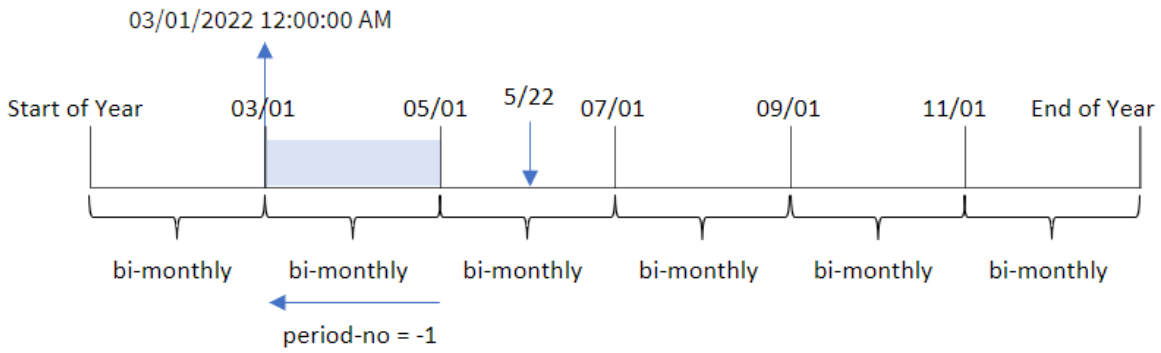
- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

Tableau de résultats

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
2/19/2022	11/01/2021	11/1/2021 12:00:00 AM
3/7/2022	01/01/2022	1/1/2022 12:00:00 AM
3/30/2022	01/01/2022	1/1/2022 12:00:00 AM
4/5/2022	01/01/2022	1/1/2022 12:00:00 AM
4/16/2022	01/01/2022	1/1/2022 12:00:00 AM
5/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/22/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	03/01/2022	3/1/2022 12:00:00 AM
6/26/2022	03/01/2022	3/1/2022 12:00:00 AM
7/9/2022	05/01/2022	5/1/2022 12:00:00 AM
7/22/2022	05/01/2022	5/1/2022 12:00:00 AM
7/23/2022	05/01/2022	5/1/2022 12:00:00 AM
7/27/2022	05/01/2022	5/1/2022 12:00:00 AM
8/2/2022	05/01/2022	5/1/2022 12:00:00 AM
8/8/2022	05/01/2022	5/1/2022 12:00:00 AM
8/19/2022	05/01/2022	5/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Si on utilise -1 comme argument `period_no` dans la fonction `monthsstart()`, après avoir initialement divisé une année en segments bimestriels, la fonction renvoie la première milliseconde du segment bimestriel précédent avant la transaction.

Diagramme de la fonction `monthsstart()`, exemple period_no



La transaction 8195 a lieu dans le segment entre mai et juin. Par conséquent, le segment bimestriel précédent était compris entre le 1er mars et le 30 avril, et la fonction renvoie la première milliseconde de ce segment, à savoir, le 1er mars 2022 à 12:00:00 AM.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `bi_monthly_start`, qui regroupe les transactions en segments bimestriels et renvoie l'horodatage de début du segment défini pour chaque transaction.

Cependant, dans cet exemple, nous devons également définir avril comme le premier mois de l'exercice financier.

Script de chargement

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

Transactions:
    Load
        *,
        monthsstart(2,date,0,4) as bi_monthly_start,
        timestamp(monthsstart(2,date,0,4)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
```

```
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

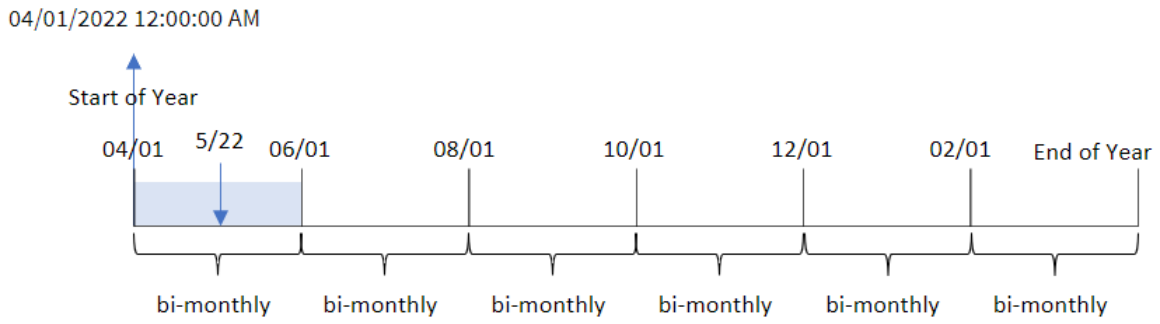
Tableau de résultats

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Si on utilise 4 comme argument `first_month_of_year` dans la fonction `monthsstart()`, la fonction commence l'année le 1er avril, puis divise l'année en segments bimestriels : Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Diagramme de la fonction `monthsstart()`, exemple first_month_of_year



La transaction 8195 a eu lieu le 22 mai et tombe dans le segment entre le 1er avril et le 31 mai. Par conséquent, la fonction renvoie la première milliseconde de ce segment, à savoir, le 1er avril 2022 à 12:00:00 AM.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui regroupe les transactions en segments bimestriels et renvoie l'horodatage de début du segment défini de chaque transaction est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez les mesures suivantes :

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

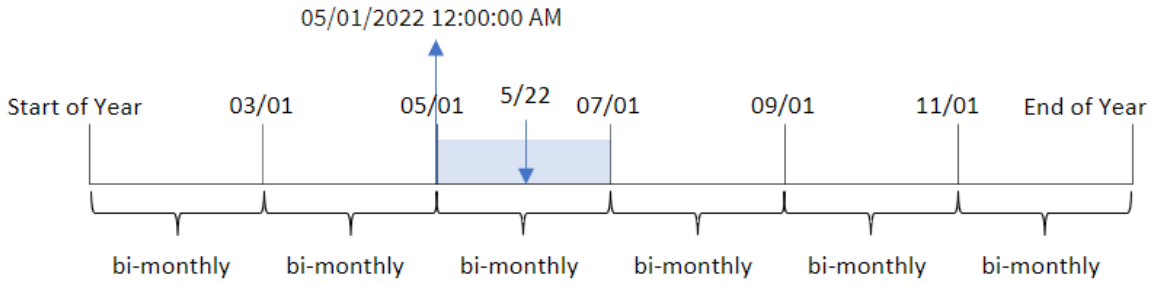
Ces calculs récupéreront l'horodatage de début du segment bimestriel de la transaction.

Tableau de résultats

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

Diagramme de la fonction `monthsstart()`, exemple objet graphique



La transaction 8195 a eu lieu le 22 mai. La fonction `monthsstart()` divise initialement l'année en segments bimestriels. La transaction 8195 tombe dans le segment entre le 1er mai et le 30 juin. Par conséquent, la fonction renvoie la première milliseconde de ce segment, à savoir, 05/01/2022 12:00:00 AM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de soldes de prêts, chargé dans une table appelée `Loans`.
- Données constituées des ID des prêts, du solde au début du mois et du taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels qui ont été accumulés pour chaque prêt pour la période de son choix. L'exercice financier commence en janvier.

Script de chargement

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

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Chargez les données et ouvrez une feuille.

Au début du script de chargement, une variable (`vPeriod`) a été créée ; elle sera liée au contrôle d'entrée de variable. Ensuite, configurez la variable comme un objet personnalisé dans la feuille.

Procédez comme suit :

1. Dans le panneau des ressources, cliquez sur **Objets personnalisés**.
2. Sélectionnez **Qlik Dashboard bundle** et créez un objet **Entrée de variable**.
3. Saisissez un titre pour l'objet graphique.

4. Sous **Variable**, sélectionnez **vPeriod** comme nom et définissez l'objet de sorte qu'il s'affiche sous forme de **Liste déroulante**.
5. Sous **Valeurs**, configurez l'objet de sorte qu'il utilise des valeurs dynamiques. Saisissez les éléments suivants :
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Ensuite, créez le tableau de résultats.

Procédez comme suit :

1. Créez un tableau. Ajoutez les champs suivants comme dimensions :
   - `employee_id`
   - `employee_name`
2. Pour calculer les intérêts accumulés, créez une mesure :
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Définissez le **Formatage des nombres** des mesures sur **Devise**. Cliquez sur **✔ Édition terminée**.
   Vous pouvez maintenant modifier les données affichées dans le tableau en ajustant le segment de temps dans l'objet variable.

Voici ce à quoi ressemblera le tableau de résultats lorsque l'option de période `month` est sélectionnée :

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

La fonction `monthsstart()`, si on utilise l'entrée de l'utilisateur comme premier argument et la date d'aujourd'hui comme deuxième argument, renvoie la date de début de la période sélectionnée par l'utilisateur. En soustrayant ce résultat de la date actuelle, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent au cours de cette période.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif encouru pour cette période. Le résultat est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts cumulés jusqu'à présent au cours de cette période.

monthstart

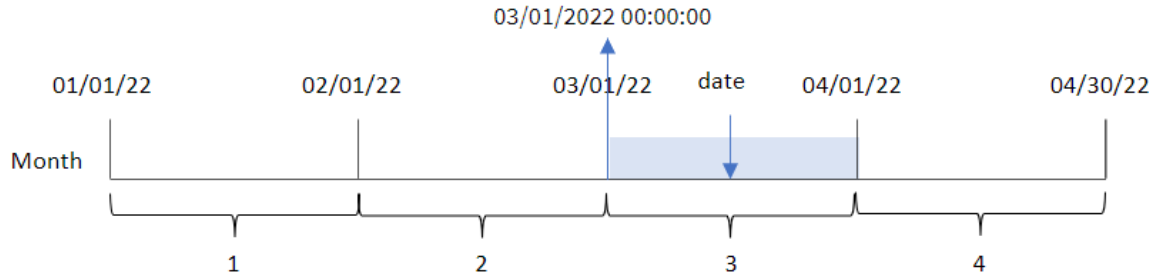
Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour du mois contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
MonthStart(date[, period_no])
```

Type de données renvoyé : double

Diagramme de la fonction `monthstart()`



La fonction `monthstart()` détermine le mois dans lequel tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la première milliseconde de ce mois-là.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier qui, s'il est égal à 0 ou s'il est omis, indique le mois contenant la **date**. Les valeurs négatives spécifiées pour **period_no** indiquent les mois passés tandis que les valeurs positives désignent les mois à venir.

Cas d'utilisation

La fonction `monthstart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction du mois qui s'est écoulée jusqu'à présent. Par exemple, elle peut être utilisée pour calculer les intérêts cumulés au cours d'un mois jusqu'à une certaine date.

Exemples de fonction

Exemple	Résultat
`monthstart('10/19/2001')`	Renvoie 10/01/2001.
`monthstart('10/19/2001', -1)`	Renvoie 09/01/2001.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `start_of_month`, qui renvoie un horodatage du début du mois au cours duquel les transactions ont eu lieu.

Script de chargement

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

Transactions:
    Load
        *,
        monthstart(date) as start_of_month,
        timestamp(monthstart(date)) as start_of_month_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
```

```
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `start_of_month`
- `start_of_month_timestamp`

Tableau de résultats

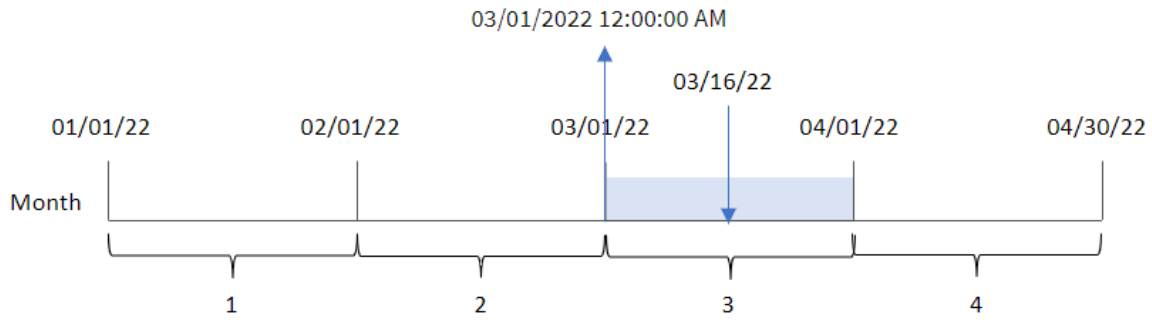
date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM

date	start_of_month	start_of_month_timestamp
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Le champ `start_of_month` est créé dans l'instruction LOAD précédente via la fonction `monthstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `monthstart()` identifie le mois dans lequel tombe la valeur date en renvoyant un horodatage pour la première milliseconde de ce mois.

Diagramme de la fonction `monthstart()`, exemple sans argument supplémentaire



La transaction 8192 a eu lieu le 16 mars. La fonction `monthstart()` renvoie la première milliseconde de ce mois, soit le 1er mars à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_month_start`, qui renvoie l'horodatage du début du mois avant la transaction.

Script de chargement

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

```
Transactions:
    Load
        *,
        monthstart(date,-1) as previous_month_start,
        timestamp(monthstart(date,-1)) as previous_month_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

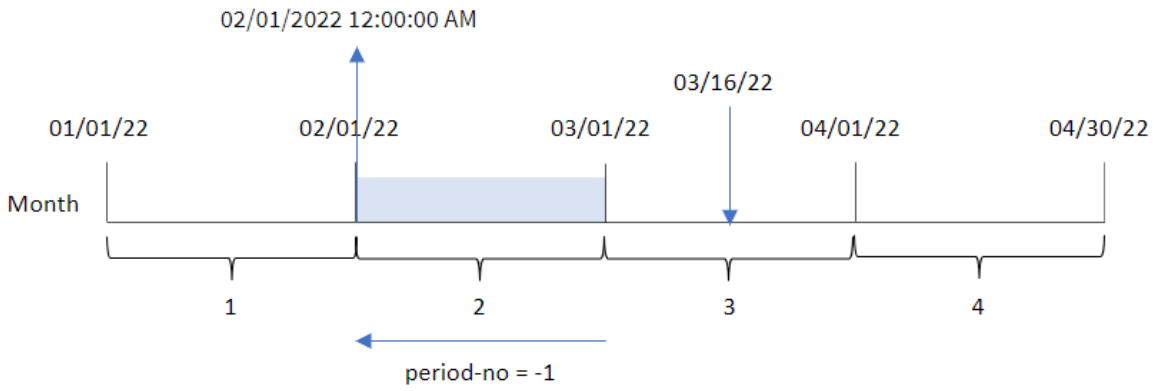
Tableau de résultats

date	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM

date	previous_month_start	previous_month_start_timestamp
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Dans cet exemple, étant donné que la valeur `period_no` de -1 a été utilisée comme argument de décalage dans la fonction `monthstart()`, la fonction commence par identifier le mois au cours duquel les transactions ont lieu. Elle décale ensuite d'un mois en arrière et identifie la première milliseconde de ce mois-là.

Diagramme de la fonction `monthstart()`, exemple period_no



La transaction 8192 a eu lieu le 16 mars. La fonction `monthstart()` identifie que le mois avant la transaction était le mois de février. Elle renvoie alors la première milliseconde de ce mois, le 1er février à 12:00:00 AM.

Exemple 3 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant au début du mois de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer la date de début du mois d'une transaction, créez les mesures suivantes :

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

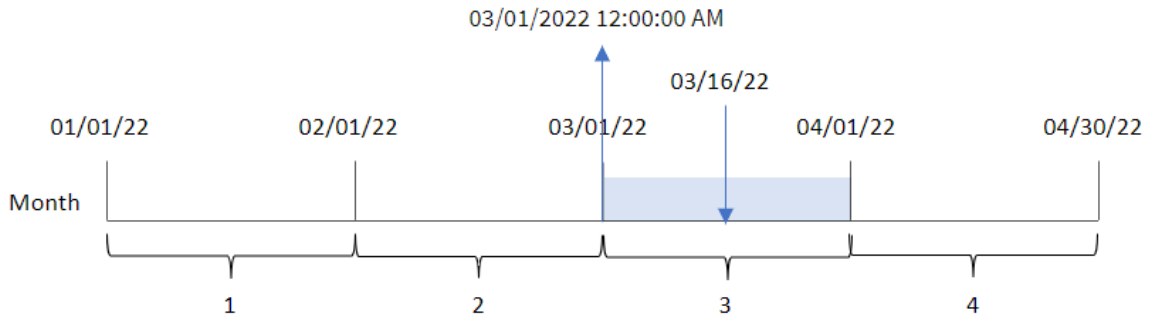
Tableau de résultats

date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

La mesure `start_of_month` est créée dans l'objet graphique via la fonction `monthstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `monthstart()` identifie le mois dans lequel tombe la valeur date en renvoyant un horodatage pour la première milliseconde de ce mois.

Diagramme de la fonction `monthstart()`, exemple objet graphique



La transaction 8192 a eu lieu le 16 mars. La fonction `monthstart()` identifie que la transaction a eu lieu en mars et renvoie la première milliseconde de ce mois, soit le 1er mars à 12:00:00 AM.

Exemple 4 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de soldes de prêts, chargé dans une table appelée `Loans`.
- Données constituées des ID des prêts, du solde au début du mois et du taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels qui ont été accumulés pour chaque prêt au cours du mois jusqu'à la date du jour.

Script de chargement

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

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :
   - `loan_id`
   - `start_balance`
2. Ensuite, pour calculer les intérêts accumulés, créez une mesure : `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Si on utilise la date d'aujourd'hui comme seul argument, la fonction `monthstart()` renvoie la date de début du mois en cours. En soustrayant ce résultat de la date actuelle, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent ce mois.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif encouru pour cette période. Le résultat est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts cumulés jusqu'à présent ce mois.

networkdays

La fonction **networkdays** renvoie le nombre de jours ouvrables (du lundi au vendredi) compris entre les valeurs **start_date** et **end_date** (incluses) en tenant compte de tous les arguments **holiday** facultatifs répertoriés.

Syntaxe :

```
networkdays (start_date, end_date [, holiday])
```

Type de données renvoyé : entier

Diagramme de calendrier affichant la plage de dates renvoyée par la fonction networkdays

La fonction `networkdays` présente les restrictions suivantes :

- Il n'existe pas de méthode pour modifier les jours ouvrables (workdays). En d'autres termes, il n'est pas possible de modifier la fonction pour les régions ou les situations pour lesquelles les jours ouvrables ne correspondent pas à la plage du lundi au vendredi.
- Le paramètre `holiday` doit être une constante de type chaîne. Les expressions ne sont pas acceptées.

Arguments

Argument	Description
start_date	Date de début à évaluer.
end_date	Date de fin à évaluer.
holiday	Périodes de congé à exclure des jours ouvrables. Un congé est indiqué sous forme de date constante de type chaîne. Vous pouvez spécifier plusieurs dates de congé si vous les séparez par des virgules. **Exemple :** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Cas d'utilisation

La fonction `networkdays()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise le nombre de jours ouvrables de la semaine qui ont lieu entre deux dates. Par exemple, si un utilisateur souhaite calculer le salaire total d'un employé en contrat PAYE (pay-as-you-earn - retenue à la source de l'impôt sur les salaires).

Exemples de fonction

Exemple	Résultat
`networkdays ('12/19/2013', '01/07/2014')`	Renvoie 14. Cet exemple ne prend pas en compte la période de congé.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Renvoie 12. Cet exemple tient compte de la période de congé allant du 12/25/2013 au 12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Renvoie 10. Cet exemple prend en compte deux périodes de congé.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des ID de projet, leurs dates de début et leurs dates de fin. Ces informations sont chargées dans une table appelée `Projects`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ supplémentaire, `net_work_days`, pour calculer le nombre de jours ouvrables impliqués dans chaque projet.

Script de chargement

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

Projects:
    Load
        *,
        networkdays(start_date,end_date) as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tableau de résultats

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	13

Étant donné qu'aucun congé n'est prévu (ils auraient été signalés dans le troisième argument de la fonction `networkdays()`), la fonction soustraie la valeur `start_date` de la valeur `end_date` ainsi que tous les weekends pour calculer le nombre de jours ouvrables entre les deux dates.

Diagramme de calendrier mettant en surbrillance les jours ouvrables du projet 5 (sans congés)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Le calendrier ci-dessus souligne visuellement le projet portant l'`id` 5. Le projet 5 commence le mercredi 10 août 2022 et se termine le 26 août 2022. Étant donné que tous les samedis et dimanches sont ignorés, cela fait un total de 13 jours ouvrables entre ces deux dates incluses.

Exemple 2 – un seul jour de congé

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux de l'exemple précédent.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ supplémentaire, `net_work_days`, pour calculer le nombre de jours ouvrables impliqués dans chaque projet.

Dans cet exemple, il existe un jour de congé prévu le 19 août 2022.

Script de chargement

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

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/19/2022') as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tableau de résultats

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Le seul jour de congé prévu est saisi comme troisième argument dans la fonction `networkdays()`.

Diagramme de calendrier mettant en surbrillance les jours ouvrables du projet 5 (un seul jour de congé)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Le calendrier ci-dessus souligne visuellement le projet 5, faisant la démonstration de cet ajustement pour inclure le jour de congé. Ce jour de congé tombe pendant le projet 5, le vendredi 19 août 2022. En conséquence, la valeur `net_work_days` totale du projet 5 diminue d'un jour et passe de 13 à 12 jours.

Exemple 3 – plusieurs jours de congé

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ supplémentaire, `net_work_days`, pour calculer le nombre de jours ouvrables impliqués dans chaque projet.

Cependant, dans cet exemple, quatre jours de congé sont prévus, du 18 août au 21 août 2022.

Script de chargement

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

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/18/2022','08/19/2022','08/20/2022','08/21/2022')
as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tableau de résultats

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	11

Les quatre jours de congé prévus sont saisis sous forme de liste séparée par des virgules à partir du troisième argument dans la fonction `networkdays()`.

Diagramme de calendrier mettant en surbrillance les jours ouvrables du projet 5 (plusieurs jours de congé)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18 Holiday	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Le calendrier ci-dessus souligne visuellement le projet 5, faisant la démonstration de cet ajustement pour inclure ces jours de congé. Cette période de congés prévus tombe pendant le projet 5, deux des jours de congé ayant lieu un jeudi et un vendredi. En conséquence, la valeur `net_work_days` totale du projet 5 passe de 13 à 11 jours.

Exemple 4 – un seul jour de congé

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

Il existe un jour de congé prévu le 19 août 2022.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le champ `net_work_days` est calculé via une mesure dans un objet graphique.

Script de chargement

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

Projects:
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `start_date`
- `end_date`

Créez la mesure suivante :

```
= networkdays(start_date,end_date,'08/19/2022')
```

Tableau de résultats

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Le seul jour de congé prévu est saisi comme troisième argument dans la fonction `networkdays()`.

Diagramme de calendrier montrant les jours ouvrables nets avec un seul jour de congé (objet graphique)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Le calendrier ci-dessus souligne visuellement le projet 5, faisant la démonstration de cet ajustement pour inclure le jour de congé. Ce jour de congé tombe pendant le projet 5, le vendredi 19 août 2022. En conséquence, la valeur `net_work_days` totale du projet 5 diminue d'un jour et passe de 13 à 12 jours.

now

Cette fonction renvoie un horodatage de l'heure actuelle. La fonction renvoie des valeurs au format de variable système **TimeStamp**. La valeur **timer_mode** par défaut est 1.

Syntaxe :

```
now([ timer_mode])
```

Type de données renvoyé : double

La fonction `now()` peut être utilisée dans le script de chargement ou dans des objets graphiques.

Arguments

Argument	Description
timer_mode	Admet les valeurs suivantes : 0 (heure du dernier chargement de données terminé) 1 (heure d'appel de la fonction) 2 (heure d'ouverture de l'application) *Si vous utilisez la fonction dans un script de chargement de données, **timer_mode=0** calcule l'heure du dernier chargement de données terminé tandis que **timer_mode=1** génère l'heure de l'appel de la fonction lors du chargement de données actif.*

Cas d'utilisation

La fonction `now()` est couramment utilisée comme composant d'une expression. Par exemple, elle peut être utilisée pour calculer le temps restant du cycle de vie d'un produit. La fonction `now()` est alors utilisée à la place de la fonction `today()` lorsque l'expression nécessite l'utilisation d'une fraction d'un jour.

Le tableau suivant explique le résultat renvoyé par la fonction `now()` suivant différentes valeurs pour l'argument `timer_mode` :

Exemples de fonction

Valeur de timer_mode	Résultat en cas d'utilisation dans un script de chargement	Résultat en cas d'utilisation dans un objet graphique
0	Renvoie un horodatage, au format de la variable système `TimeStamp`, du dernier chargement de données réussi précédant le dernier chargement de données.	Renvoie un horodatage, au format de la variable système `TimeStamp`, du dernier chargement de données.
1	Renvoie un horodatage, au format de la variable système `TimeStamp`, du dernier chargement de données.	Renvoie un horodatage, au format de la variable système `TimeStamp`, de l'appel de la fonction.
2	Renvoie un horodatage, au format de la variable système `TimeStamp`, correspondant au début de la session de l'utilisateur dans l'application. Cette valeur ne sera pas mise à jour, sauf si l'utilisateur charge de nouveau le script.	Renvoie l'horodatage, au format de la variable système `TimeStamp`, correspondant au début de la session de l'utilisateur dans l'application. Cette valeur sera actualisée une fois qu'une nouvelle session commencera ou lorsque les données de l'application seront de nouveau chargées.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – génération d'objets via un script de chargement

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Cet exemple crée trois variables avec la fonction `now()`. Chaque variable utilise l'une des options `timer_mode` pour démontrer leur effet.

Pour que les variables démontrent leur objectif, chargez le script, puis, après une courte période, chargez-le une deuxième fois. Cela a pour résultat d'afficher des valeurs différentes dans les variables `now(0)` et `now(1)`, ce qui permet de démontrer correctement leur objectif.

Script de chargement

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Résultats

Une fois les données chargées pour la deuxième fois, créez trois zones de texte en suivant les instructions ci-dessous.

Commencez par créer une zone de texte pour les données précédemment chargées.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vPreviousDataLoad`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Previous Reload Time' à l'objet.

Créez ensuite une zone de texte pour les données en cours de chargement.

Procédez comme suit :

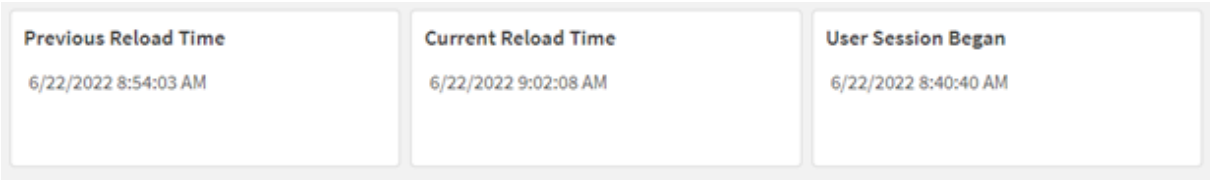
1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vCurrentDataLoad`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Current Reload Time' à l'objet.

Créez une dernière zone de texte à afficher lors du démarrage de la session de l'utilisateur dans l'application.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vApplicationOpened`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'User Session Started' à l'objet.

Variables de script de chargement `now()`



L'image ci-dessus montre des exemples de valeur pour chacune des variables créées. Par exemple, les valeurs pourraient être comme suit :

- Heure du chargement précédent : 6/22/2022 8:54:03 AM
- Heure du chargement actuel : 6/22/2022 9:02:08 AM
- Début de la session utilisateur : 6/22/2022 8:40:40 AM

Exemple 2 – génération d'objets sans script de chargement

Script de chargement et expression de graphique

Vue d'ensemble

Dans cet exemple, vous allez créer trois objets graphiques utilisant la fonction `now()` sans chargement de variables ni de données dans l'application. Chaque objet graphique utilise l'une des options `timer_mode` pour démontrer leur effet.

Pour cet exemple, il n'existe aucun script de chargement.

Procédez comme suit :

1. Ouvrez l'éditeur de chargement de données.
2. Sans modifier le script de chargement existant, cliquez sur **Charger les données**.
3. Peu de temps après, chargez le script une deuxième fois.

Résultats

Une fois les données chargées pour la deuxième fois, créez trois zones de texte.

Commencez par créer une zone de texte pour le dernier chargement de données.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :
   ```
   =now(0)
   ```
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Dernier chargement de données' à l'objet.

Créez ensuite une zone de texte affichant l'heure actuelle.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :
   ```
   =now(1)
   ```
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Heure actuelle' à l'objet.

Créez une dernière zone de texte à afficher lors du démarrage de la session de l'utilisateur dans l'application.

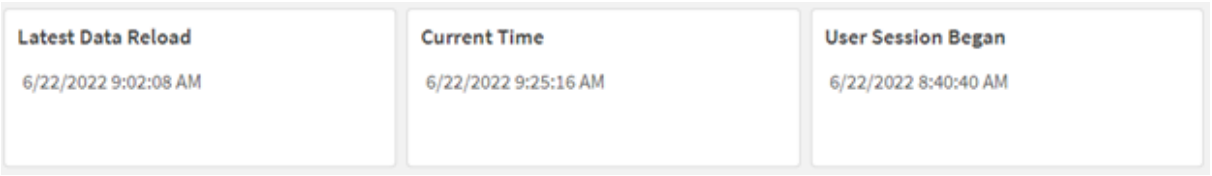
Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :

```
=now(2)
```

3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Début de la session utilisateur' à l'objet.

Exemples d'objet graphique `now()`



L'image ci-dessus montre des exemples de valeur pour chacun des objets créés. Par exemple, les valeurs pourraient être comme suit :

- Dernier chargement de données : 6/22/2022 9:02:08 AM
- Heure actuelle : 6/22/2022 9:25:16 AM
- Début de la session utilisateur : 6/22/2022 8:40:40 AM

L'objet graphique 'Dernier chargement de données' utilise une valeur `timer_mode` égale à 0. Cela renvoie l'horodatage du dernier chargement de données réussi.

L'objet graphique 'Date actuelle' utilise une valeur `timer_mode` égale à 1. Cela renvoie la date actuelle conformément à l'horloge système. Si la feuille ou l'objet est actualisé, cette valeur sera mise à jour.

L'objet graphique 'Début de la session utilisateur' utilise une valeur `timer_mode` égale à 2. Cela renvoie l'horodatage de l'ouverture de l'application et du début de la session utilisateur.

Exemple 3 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données constitué de l'inventaire d'une opération de minage de cryptomonnaie, chargé dans une table appelée `Inventory`.
- Données avec les champs suivants : `id`, `purchase_date` et `wph` (watts par heure).

L'utilisateur souhaite un tableau qui affiche, par `id`, le coût total encouru par chaque rig de minage au cours du mois jusqu'à ce jour, en termes de consommation d'énergie.

Cette valeur doit se mettre à jour à chaque actualisation de l'objet graphique. Le coût actuel de l'électricité est de 0,0678 $ parr kWH.

Script de chargement

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

Inventory:
Load
*
Inline
[
id,purchase_date,wph
8188,1/7/2022,1123
8189,1/19/2022,1432
8190,2/28/2022,1227
8191,2/5/2022,1322
8192,3/16/2022,1273
8193,4/1/2022,1123
8194,5/7/2022,1342
8195,5/16/2022,2342
8196,6/15/2022,1231
8197,6/26/2022,1231
8198,7/9/2022,1123
8199,7/22/2022,1212
8200,7/23/2022,1223
8201,7/27/2022,1232
8202,8/2/2022,1232
8203,8/8/2022,1211
8204,8/19/2022,1243
8205,9/26/2022,1322
8206,10/14/2022,1133
8207,10/29/2022,1231
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `id`.

Créez la mesure suivante :

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Si l'objet graphique a été actualisé le 6/22/2022 à 10:39:05 AM, il renvoie les résultats suivants :

Tableau de résultats

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	$46.13
8192	$44.42

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95
8197	$42.95
8198	$39.18
8199	$42.29
8200	$42.67
8201	$42.99
8202	$42.99
8203	$42.25
8204	$43.37
8205	$46.13
8206	$39.53

L'utilisateur souhaite que les résultats de l'objet soient actualisés à chaque actualisation de l'objet. D'où l'argument `timer_mode` fourni pour les instances de la fonction `now()` dans l'expression. L'horodatage du début du mois, identifié en utilisant la fonction `now()` comme argument d'horodatage de la fonction `monthstart()`, est soustrait de l'heure actuelle identifiée par la fonction `now()`. Cela donne le temps total écoulé jusqu'ici pour ce mois, en jours.

Cette valeur est multipliée par 24 (le nombre d'heures dans une journée), puis par la valeur du champ `wph`.

Pour convertir les watts par heure en kilowatts par heure, le résultat est divisé par 1 000 avant de finir par être multiplié par le tarif de kWH appliqué.

quarterend

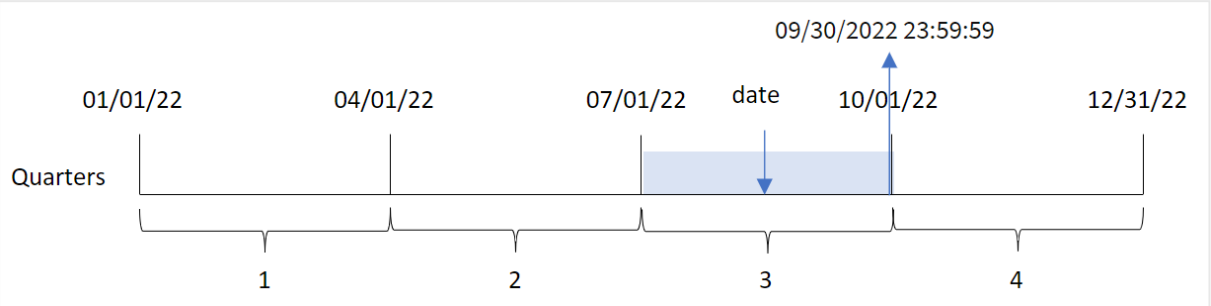
Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du trimestre contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction `quarterend()`



La fonction `quarterend()` détermine le trimestre dans lequel tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la dernière milliseconde du dernier mois de ce trimestre-là. Le premier mois de l'année est, par défaut, janvier. Cependant, vous pouvez modifier le mois défini comme le premier en utilisant l'argument `first_month_of_year` dans la fonction `quarterend()`.

> *La fonction `quarterend()` ne tient pas compte de la variable système `FirstMonthOfYear`. L'année commence le 1er janvier, sauf si l'argument `first_month_of_year` est utilisé pour modifier cela.*

Cas d'utilisation

La fonction `quarterend()` est couramment utilisée dans le cadre d'une expression lorsque vous souhaitez que le calcul utilise la fraction du trimestre qui n'a pas encore eu lieu. Par exemple, si vous souhaitez calculer le total des intérêts non encore encourus au cours du trimestre.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique le trimestre comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les trimestres passés tandis que les valeurs positives désignent les trimestres à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`quarterend('10/29/2005')`	Returns 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Returns 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Returns 11/30/2005 23:59:59.

Exemple 1 - exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent contenant les éléments suivants :
    - La fonction `quarterend()` définie comme le champ '`end_of_quarter`' et qui renvoie un horodatage de la fin du trimestre de réalisation des transactions.
    - La fonction `timestamp()` définie comme le champ '`end_of_quarter_timestamp`' et qui renvoie l'horodatage exact de la fin du trimestre sélectionné.

Script de chargement

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

Transactions:
    Load
       *,
       quarterend(date) as end_of_quarter,
       timestamp(quarterend(date)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
```

```
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

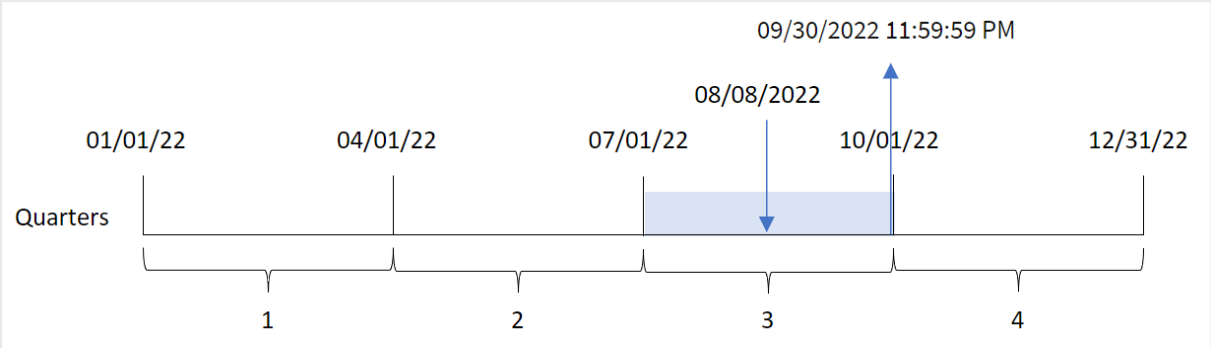
Tableau de résultats

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Le champ '`end_of_quarter`' est créé dans l'instruction LOAD précédente via la fonction `quarterend()` et en transmettant le champ date comme argument de la fonction.

La fonction `quarterend()` identifie initialement le trimestre dans lequel tombe la valeur date et renvoie un horodatage pour la dernière milliseconde de ce trimestre.

Diagramme de la fonction `quarterend()` avec la fin du trimestre de la transaction 8203 identifiée



La transaction 8203 a eu lieu le 8 août. La fonction `quarterend()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie la dernière milliseconde de ce trimestre-là, à savoir, le 30 septembre à 11:59:59 PM.

Exemple 2 - period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent contenant les éléments suivants :
  - La fonction `quarterend()` définie comme le champ '`previous_quarter_end`' et qui renvoie un horodatage de la fin du trimestre avant la réalisation de la transaction.
  - La fonction `timestamp()` définie comme le champ '`previous_end_of_quarter_timestamp`' et qui renvoie l'horodatage exact de la fin du trimestre avant la réalisation de la transaction.

Script de chargement

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

Transactions:
    Load
       *,
       quarterend(date, -1) as previous_quarter_end,
       timestamp(quarterend(date, -1)) as previous_quarter_end_timestamp
       ;
Load
*
```

```
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- previous_quarter_end
- previous_quarter_end_timestamp

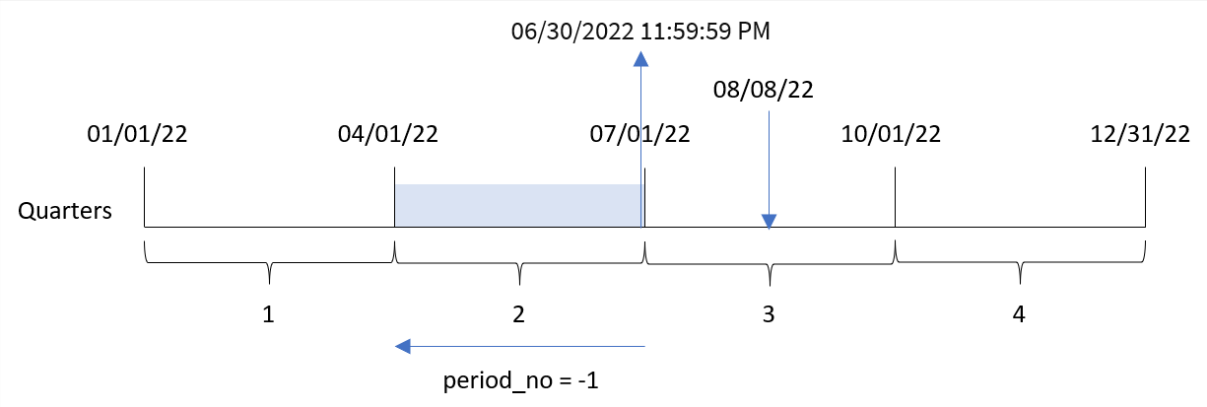
Tableau de résultats

id	date	previous_quarter_end	previous_quarter_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	12/31/2021	12/31/2021 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8195	5/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8196	6/15/2022	03/31/2022	3/31/2022 11:59:59 PM

id	date	previous_quarter_end	previous_quarter_end_timestamp
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Étant donné qu'un argument `period_no` de `-1` est utilisé comme argument de décalage dans la fonction `quarterend()`, la fonction commence par identifier le trimestre au cours duquel les transactions ont lieu. Elle décale ensuite d'un trimestre en arrière et identifie la dernière milliseconde de ce trimestre-là.

Diagramme de la fonction `quarterend()` avec une valeur period_no de -1



La transaction 8203 a eu lieu le 8 août. La fonction `quarterend()` identifie que le trimestre avant la transaction était compris entre le 1er avril et le 30 juin. La fonction renvoie alors la dernière milliseconde de ce trimestre, à savoir, le 30 juin à 11:59:59 PM.

Exemple 3 - first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions en 2022, chargé dans une table appelée '`Transactions`'.
- Chargement précédent contenant les éléments suivants :
    - La fonction `quarterend()` définie comme le champ '`end_of_quarter`' et qui renvoie un horodatage de la fin du trimestre de réalisation des transactions.
    - La fonction `timestamp()` définie comme le champ '`end_of_quarter_timestamp`' et qui renvoie l'horodatage exact de la fin du trimestre sélectionné.

Cependant, dans cet exemple, la stratégie de l'entreprise détermine que l'exercice financier commence le 1er mars.

Script de chargement

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

Transactions:
    Load
       *,
       quarterend(date, 0, 3) as end_of_quarter,
       timestamp(quarterend(date, 0, 3)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
```

```
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

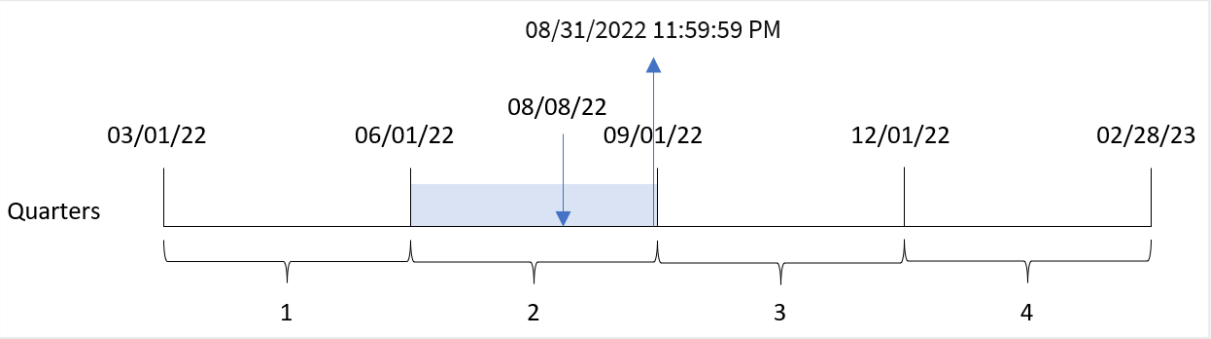
Résultats

Tableau de résultats

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Étant donné que l'argument `first_month_of_year` égal à `3` est utilisé dans la fonction `quarterend()`, il reporte le début de l'année du 1er janvier au 1er mars.

Diagramme de la fonction `quarterend()` avec mars comme premier mois de l'année



La transaction 8203 a eu lieu le 8 août. Étant donné que le début de l'année est le 1er mars, les trimestres de l'année sont les suivants : Mar-May, Jun-Aug, Sep-Nov et Dec-Feb.

La fonction `quarterend()` identifie que la transaction a eu lieu au cours du trimestre compris entre le début de juin et d'août et renvoie la dernière milliseconde de ce trimestre-là, à savoir, le 31 août à 11:59:59 PM.

Exemple 4 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui renvoie un horodatage correspondant à la fin du trimestre de réalisation des transactions est créé sous forme de mesure dans un graphique de l'application.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`

Pour calculer la date de fin du trimestre d'une transaction, créez les mesures suivantes :

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Tableau de résultats

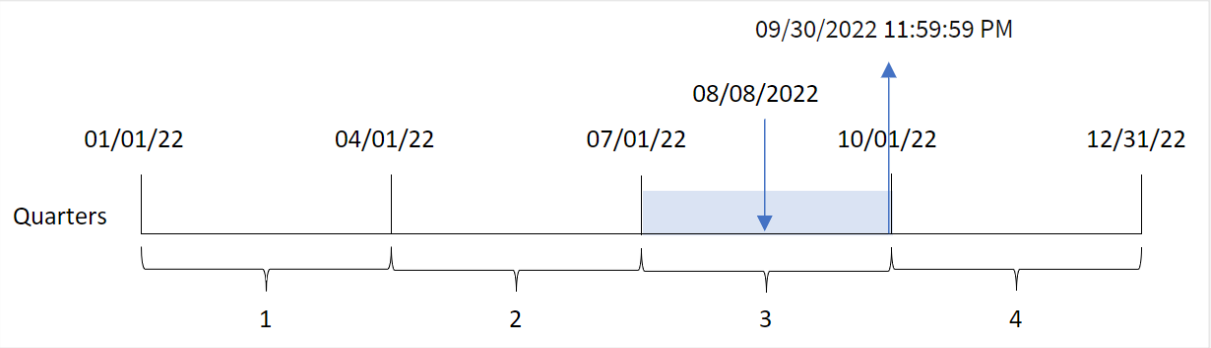
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	=quarterend(date)	=timestamp(quarterend(date))
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Le champ '`end_of_quarter`' est créé dans l'instruction LOAD précédente via la fonction `quarterend()` et en transmettant le champ date comme argument de la fonction.

La fonction `quarterend()` identifie initialement le trimestre dans lequel tombe la valeur date et renvoie un horodatage pour la dernière milliseconde de ce trimestre.

Diagramme de la fonction `quarterend()` avec la fin du trimestre de la transaction 8203 identifiée



La transaction 8203 a eu lieu le 8 août. La fonction `quarterend()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie la dernière milliseconde de ce trimestre-là, à savoir, le 30 septembre à 11:59:59 PM.

Exemple 5 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Employee_Expenses`'. La table contient les champs suivants :
  - ID des employés
  - Nom des employés
  - Notes de frais quotidiennes moyennes de chaque employé

L'utilisateur final souhaite un objet graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées qu'il reste à encourir pour le reste du trimestre. L'exercice financier commence en janvier.

Script de chargement

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `employee_id`
- `employee_name`

Pour calculer les intérêts accumulés, créez la mesure suivante :

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

employee_id	employee_name	=(quarterend(today(1))-today(1))*avg_daily_claim
182	Mark	$480.00
183	Deryck	$400.00
184	Dexter	$400.00
185	Sydney	$864.00
186	Agatha	$576.00

La fonction `quarterend()` utilise la date d'aujourd'hui comme seul argument et renvoie la date de fin du mois en cours. Elle soustraie ensuite la date d'aujourd'hui de la date de fin de l'année et l'expression renvoie le nombre de jours qu'il reste dans ce mois.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes de chaque employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours du trimestre restant.

quartername

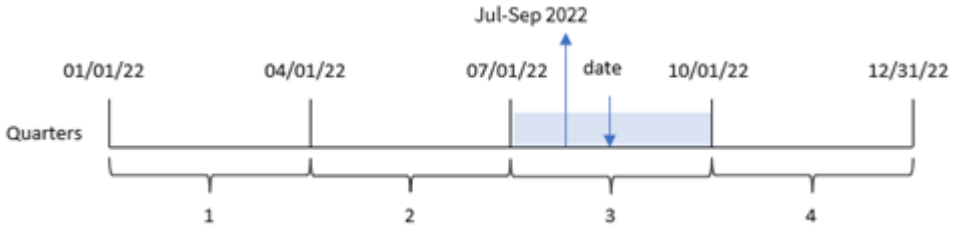
Cette fonction renvoie une valeur d'affichage présentant les mois du trimestre (formatés selon la variable de script **MonthNames**) et l'année avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour du trimestre.

Syntaxe :

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction `quartername()`



La fonction `quartername()` détermine le trimestre dans lequel tombe la date. Elle renvoie ensuite une valeur indiquant les mois de début et de fin de ce trimestre ainsi que l'année. La valeur numérique sous-jacente de ce résultat est la première milliseconde du trimestre.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique le trimestre comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les trimestres passés tandis que les valeurs positives désignent les trimestres à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `quartername()` est utile lorsque vous souhaitez comparer des agrégations par trimestre. Par exemple, si vous souhaitez voir les ventes totales de produits par trimestre.

Cette fonction peut être utilisée dans le script de chargement pour créer un champ dans une table Calendrier principal. Sinon, elle peut également être directement utilisée dans un graphique comme dimension calculée.

Ces exemples utilisent le format de date MM/DD/YYYY. Le format de date est indiqué dans l'instruction `SET DateFormat` située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de fonction

Exemple	Résultat
`quartername('10/29/2013')`	Renvoie Oct-Dec 2013.
`quartername('10/29/2013', -1)`	Renvoie Jul-Sep 2013.
`quartername('10/29/2013', 0, 3)`	Renvoie Sep-Nov 2013.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – date sans aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `transaction_quarter`, qui renvoie le trimestre au cours duquel les transactions ont eu lieu.

Ajoutez un autre texte ici, si nécessaire, avec des listes, etc.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
```

```
    Load
        *,
        quartername(date) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `transaction_quarter`

Tableau de résultats

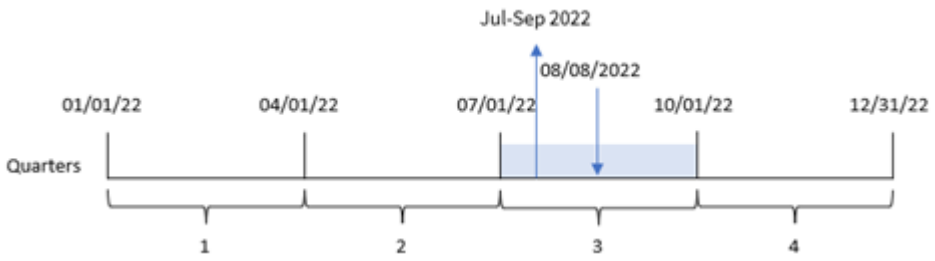
date	transaction_quarter
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022

date	transaction_quarter
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Le champ `transaction_quarter` est créé dans l'instruction LOAD précédente via la fonction `quartername()` et en transmettant le champ date comme argument de la fonction.

La fonction `quartername()` identifie initialement le trimestre au cours duquel tombe la valeur date. Elle renvoie ensuite une valeur indiquant les mois de début et de fin de ce trimestre, ainsi que l'année.

Diagramme de la fonction `quartername()`, exemple sans argument supplémentaire



La transaction 8203 a eu lieu le 8 août 2022. La fonction `quartername()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie par conséquent Jul-Sep 2022. Les mois sont affichés dans le même format que celui de la variable système `MonthNames`.

Exemple 2 – date avec argument period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_quarter`, qui renvoie le trimestre précédent avant la réalisation des transactions.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,-1) as previous_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

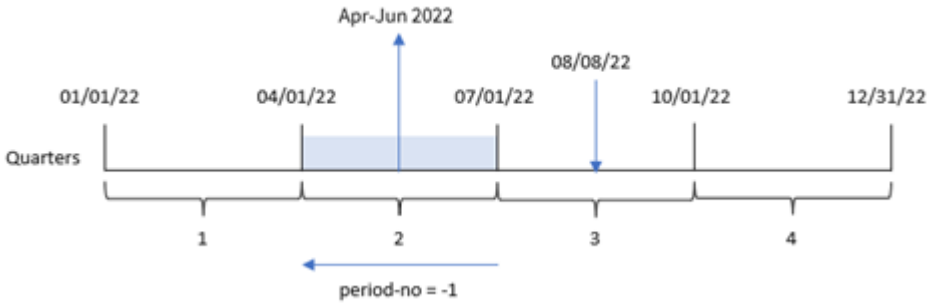
- `date`
- `previous_quarter`

Tableau de résultats

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021
2/28/2022	Oct-Dec 2021
3/16/2022	Oct-Dec 2021
4/1/2022	Jan-Mar 2022
5/7/2022	Jan-Mar 2022
5/16/2022	Jan-Mar 2022
6/15/2022	Jan-Mar 2022
6/26/2022	Jan-Mar 2022
7/9/2022	Apr-Jun 2022
7/22/2022	Apr-Jun 2022
7/23/2022	Apr-Jun 2022
7/27/2022	Apr-Jun 2022
8/2/2022	Apr-Jun 2022
8/8/2022	Apr-Jun 2022
8/19/2022	Apr-Jun 2022
9/26/2022	Apr-Jun 2022
10/14/2022	Jul-Sep 2022
10/29/2022	Jul-Sep 2022

Dans cet exemple, étant donné que l'argument `period_no` égal à -1 a été utilisé comme argument de décalage dans la fonction `quartername()`, la fonction commence par identifier que les transactions ont eu lieu au cours du troisième trimestre. Elle décale ensuite d'un trimestre en arrière et renvoie une valeur indiquant les mois de début et de fin de ce trimestre, ainsi que l'année.

Diagramme de la fonction `quartername()`, exemple period_no



La transaction 8203 a eu lieu le 8 août. La fonction `quartername()` identifie que le trimestre avant la transaction était compris entre le 1er avril et le 30 juin. Par conséquent, elle renvoie Apr-Jun 2022.

Exemple 3 – date avec argument first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, nous devons définir le 1er mars comme le début de l'exercice financier.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,0,3) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
```

```
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `transaction_quarter`

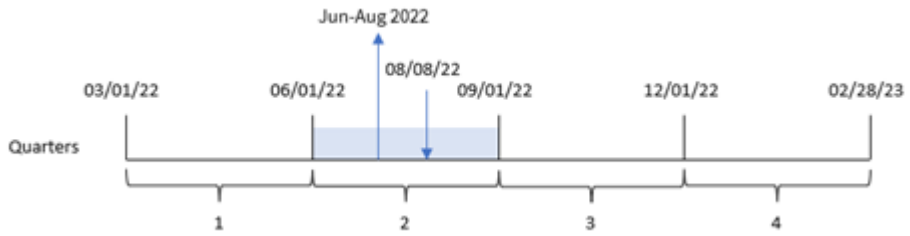
Tableau de résultats

date	transaction_quarter
1/7/2022	Dec-Feb 2021
1/19/2022	Dec-Feb 2021
2/5/2022	Dec-Feb 2021
2/28/2022	Dec-Feb 2021
3/16/2022	Mar-May 2022
4/1/2022	Mar-May 2022
5/7/2022	Mar-May 2022
5/16/2022	Mar-May 2022
6/15/2022	Jun-Aug 2022
6/26/2022	Jun-Aug 2022
7/9/2022	Jun-Aug 2022
7/22/2022	Jun-Aug 2022
7/23/2022	Jun-Aug 2022
7/27/2022	Jun-Aug 2022
8/2/2022	Jun-Aug 2022
8/8/2022	Jun-Aug 2022
8/19/2022	Jun-Aug 2022

date	transaction_quarter
9/26/2022	Sep-Nov 2022
10/14/2022	Sep-Nov 2022
10/29/2022	Sep-Nov 2022

Dans cet exemple, étant donné que l'argument `first_month_of_year` égal à 3 est utilisé dans la fonction `quartername()`, le début de l'année passe du 1er janvier au 1er mars. Par conséquent, les trimestres de l'année deviennent March-May, June-August, September-November et December-February.

Diagramme de la fonction `quartername()`, exemple first_week_day



La transaction 8203 a eu lieu le 8 août. La fonction `quartername()` identifie que la transaction a eu lieu au cours du deuxième trimestre, entre le début du mois de juin et la fin du mois d'août. Par conséquent, elle renvoie Jun-Aug 2022.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant à la fin du trimestre de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Créez la mesure suivante :

```
=quartername(date)
```

Tableau des résultats

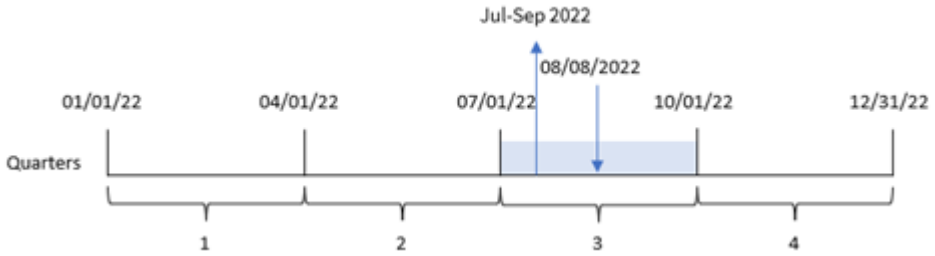
date	=quartername(date)
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022

date	=quartername(date)
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

La mesure `transaction_quarter` est créée dans l'objet graphique via la fonction `quartername()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `quartername()` identifie initialement le trimestre au cours duquel tombe la valeur date. Elle renvoie ensuite une valeur indiquant les mois de début et de fin de ce trimestre, ainsi que l'année.

Diagramme de la fonction `quartername()`, exemple objet graphique



La transaction 8203 a eu lieu le 8 août 2022. La fonction `quartername()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie par conséquent Jul-Sep 2022. Les mois sont affichés dans le même format que celui de la variable système `MonthNames`.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.

- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

L'utilisateur final souhaite un objet graphique présentant les ventes totales par trimestre pour les transactions. Cela est possible même lorsque cette dimension n'est pas disponible dans le modèle de données, en utilisant la fonction `quartername()` comme dimension calculée dans le graphique.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez une dimension calculée à l'aide de l'expression suivante :
   `=quartername(date)`
3. Calculez ensuite les ventes totales via la mesure d'agrégation suivante :
   `=sum(amount)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

=quartername(date)	=sum(amount)
Jul-Sep 2022	$446.31
Apr-Jun 2022	$351.48
Jan-Mar 2022	$253.89
Oct-Dec 2022	$163.91

quarterstart

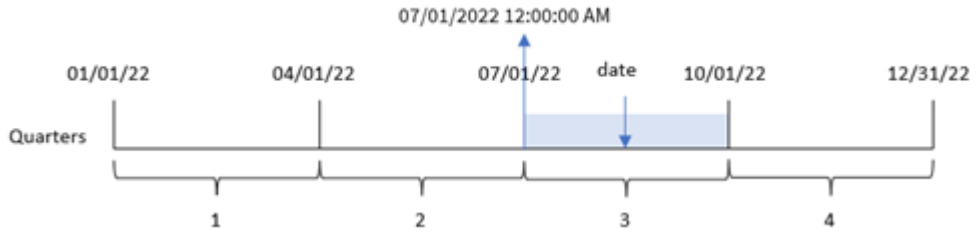
Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du trimestre contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Type de données renvoyé : double

Diagramme de la fonction `quarterstart()`



La fonction `quarterstart()` détermine le trimestre dans lequel tombe la `date`. Elle renvoie ensuite un horodatage, au format date, pour la première milliseconde du premier mois de ce trimestre-là.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique le trimestre comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les trimestres passés tandis que les valeurs positives désignent les trimestres à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Cas d'utilisation

La fonction `quarterstart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction du trimestre qui s'est écoulée jusqu'à présent. Par exemple, elle peut être utilisée si un utilisateur souhaite calculer les intérêts cumulés au cours d'un trimestre jusqu'à la date d'aujourd'hui.

Exemples de fonction

Exemple	Résultat
`quarterstart('10/29/2005')`	Renvoie 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Renvoie 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Renvoie 09/01/2005.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `start_of_quarter`, qui renvoie un horodatage du début du trimestre au cours duquel les transactions ont eu lieu.

Script de chargement

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

Transactions:
    Load
        *,
        quarterstart(date) as start_of_quarter,
        timestamp(quarterstart(date)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

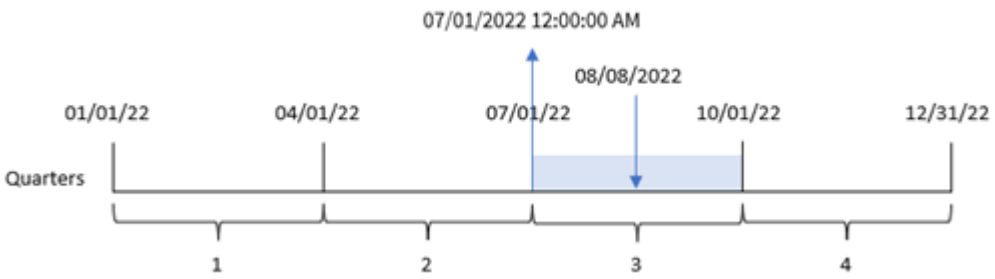
Tableau de résultats

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2021 12:00:00 AM
5/7/2022	04/01/2022	4/1/2021 12:00:00 AM
5/16/2022	04/01/2022	4/1/2021 12:00:00 AM
6/15/2022	04/01/2022	4/1/2021 12:00:00 AM
6/26/2022	04/01/2022	4/1/2021 12:00:00 AM
7/9/2022	07/01/2022	7/1/2021 12:00:00 AM
7/22/2022	07/01/2022	7/1/2021 12:00:00 AM
7/23/2022	07/01/2022	7/1/2021 12:00:00 AM
7/27/2022	07/01/2022	7/1/2021 12:00:00 AM
8/2/2022	07/01/2022	7/1/2021 12:00:00 AM
8/8/2022	07/01/2022	7/1/2021 12:00:00 AM
8/19/2022	07/01/2022	7/1/2021 12:00:00 AM
9/26/2022	07/01/2022	7/1/2021 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Le champ `start_of_quarter` est créé dans l'instruction LOAD précédente via la fonction `quarterstart()` et en transmettant le champ date comme argument de la fonction. La fonction `quarterstart()` identifie initialement le trimestre au cours duquel tombe la valeur date. Elle renvoie ensuite un horodatage de la première milliseconde de ce trimestre-là.

Diagramme de la fonction `quarterstart()`, exemple sans argument supplémentaire



La transaction 8203 a eu lieu le 8 août. La fonction `quarterstart()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie la première milliseconde de ce trimestre-là, à savoir, le 1er juillet à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_quarter_start`, qui renvoie l'horodatage du début du trimestre avant la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        quarterstart(date,-1) as previous_quarter_start,
        timestamp(quarterstart(date,-1)) as previous_quarter_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

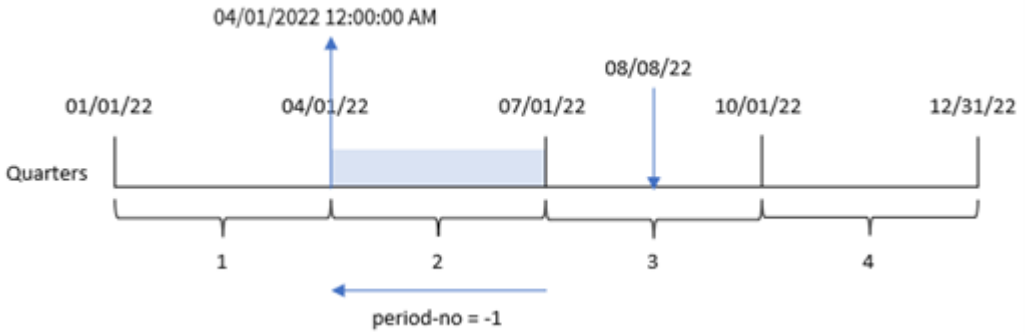
- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

Tableau de résultats

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM
8/19/2022	04/01/2022	4/1/2021 12:00:00 AM
9/26/2022	04/01/2022	4/1/2021 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Dans cet exemple, étant donné que l'argument `period_no` de -1 a été utilisé comme argument de décalage dans la fonction `quarterstart()`, la fonction commence par identifier le trimestre au cours duquel les transactions ont lieu. Elle décale ensuite d'un trimestre en arrière et identifie la première milliseconde de ce trimestre-là.

Diagramme de la fonction `quarterstart()`, exemple period_no



La transaction 8203 a eu lieu le 8 août. La fonction `quarterstart()` identifie que le trimestre avant la transaction était compris entre le 1er avril et le 30 juin. Elle renvoie alors la première milliseconde de ce trimestre, le 1er avril à 12:00:00 AM.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, nous devons définir le 1er mars comme le début de l'exercice financier.

Script de chargement

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

Transactions:
    Load
        *,
        quarterstart(date,0,3) as start_of_quarter,
        timestamp(quarterstart(date,0,3)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

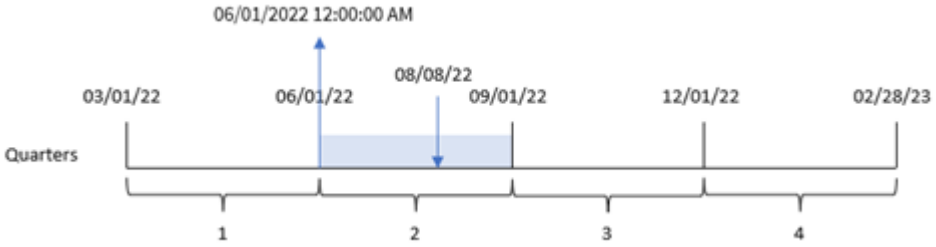
Tableau de résultats

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	06/01/2022	6/1/2022 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
8/8/2022	06/01/2022	6/1/2022 12:00:00 AM
8/19/2022	06/01/2022	6/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Dans cet exemple, étant donné que l'argument `first_month_of_year` égal à 3 est utilisé dans la fonction `quarterstart()`, il reporte le début de l'année du 1er janvier au 1er mars.

Diagramme de la fonction `quarterstart()`, exemple first_month_of_year



La transaction 8203 a eu lieu le 8 août. Étant donné que le début de l'année est le 1er mars, les trimestres de l'année sont les suivants : March-May, June-August, September-November et December-February. La fonction `quarterstart()` identifie que la transaction a eu lieu au cours du trimestre compris entre le début de juin et d'août et renvoie la première milliseconde de ce trimestre-là, à savoir, le 1er juin à 12:00:00 AM.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant à la fin du trimestre de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
```

```
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Ajoutez les mesures suivantes :

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

Tableau de résultats

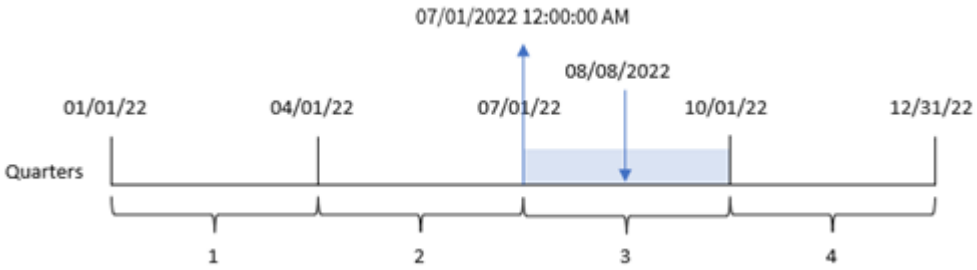
date	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM

date	=quarterstart(date)	=timestamp(quarterstart(date))
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	04/01/2022	4/1/2022 12:00:00 AM
6/26/2022	04/01/2022	4/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM

La mesure `start_of_quarter` est créée dans l'objet graphique via la fonction `quarterstart()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `quarterstart()` identifie le trimestre de la valeur date en renvoyant un horodatage pour la première milliseconde de ce trimestre-là.

Diagramme de la fonction `quarterstart()`, exemple objet graphique



La transaction 8203 a eu lieu le 8 août. La fonction `quarterstart()` identifie que la transaction a eu lieu au cours du troisième trimestre et renvoie la première milliseconde de ce trimestre-là. La valeur renvoyée est le 1er juillet à 12:00:00 AM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de soldes de prêts, chargé dans une table appelée `Loans`.
- Données constituées des ID des prêts, du solde au début du trimestre et du taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels qui ont été accumulés pour chaque prêt au cours du trimestre jusqu'à la date du jour.

Script de chargement

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :
    - `loan_id`
    - `start_balance`
2. Ensuite, pour calculer les intérêts accumulés, créez la mesure suivante :
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Si on utilise la date d'aujourd'hui comme seul argument, la fonction `quarterstart()` renvoie la date de début de l'année en cours. Si on soustraie ce résultat de la date actuelle, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent ce trimestre.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif encouru pour cette période. Le résultat est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts cumulés jusqu'à présent ce trimestre.

second

Cette fonction renvoie un entier représentant la seconde au cours de laquelle la fraction de l'**expression** est interprétée comme une heure selon l'interprétation standard des nombres.

Syntaxe :

```
second (expression)
```

Type de données renvoyé : entier

Cas d'utilisation

La fonction `second()` est utile lorsque vous souhaitez comparer des agrégations par seconde. Par exemple, la fonction peut être utilisée si vous souhaitez afficher la répartition du nombre d'activités par seconde.

Ces dimensions peuvent être créées soit dans le script de chargement, en utilisant la fonction pour créer un champ dans une table Master Calendar, soit directement dans un graphique sous forme de dimension calculée.

Exemples de fonction

Exemple	Résultat
`second( '09:14:36' )`	renvoie 36.
`second( '0.5555' )`	renvoie 55 (car 0.5555 = 13:19:55).

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 - variable

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant des transactions par horodatage chargé dans une table appelée `Transactions`.
- La variable système `TimeStamp` par défaut (`M/D/YYYY h:mm:ss[.fff] TT`) est utilisée.
- Création d'un champ, `second`, pour calculer la date des achats.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        second(date) as second
        ;
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `second`

Tableau de résultats

date	second
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Les valeurs du champ `second` sont créées via la fonction `second()` et en transmettant la date comme expression dans l'instruction LOAD précédente.

Exemple 2 – objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Les valeurs `second` sont calculées via une mesure dans un objet graphique.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
```

```
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :`date`.

Créez la mesure suivante :

```
=second(date)
```

Tableau des résultats

date	=second(date)
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Les valeurs de `second` sont créées via la fonction `second()` et en transmettant la date comme expression dans une mesure de l'objet graphique.

Exemple 3 – scénario

Script de chargement et expressions de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données d'horodatages, généré pour représenter le trafic d'un site Web de vente de billets pour un festival donné. Ces horodatages et une variable `id` correspondante sont chargés

dans une table appelée `Web_Traffic`.

- La variable système `TimeStamp M/D/YYYY h:mm:ss[.fff] TT` est utilisée.

Dans ce scénario, on comptait 10 000 billets, mis en vente à 9h00 le 20 mai 2021. Une minute plus tard, les billets étaient épuisés.

L'utilisateur souhaite un objet graphique qui montre, par seconde, le nombre de visites du site Web.

Script de chargement

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
load
    makedate(2022,05,20) as date
AutoGenerate 1;

join load
    maketime(9+floor(rand()*2),0,floor(rand()*59)) as time
autogenerate 10000;

Web_Traffic:
load
    recno() as id,
    timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Ensuite, créez une dimension calculée à l'aide de l'expression suivante :
   `=second(timestamp)`
3. Créez une mesure d'agrégation pour calculer le nombre total d'entrées :
   `=count(id)`

Le tableau de résultats ressemblera au tableau ci-dessous, mais avec des valeurs différentes pour la mesure d'agrégation :

Tableau de résultats

second(timestamp)	=count(id)
0	150
1	184
2	163

second(timestamp)	=count(id)
3	178
4	179
5	158
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
+39 lignes supplémentaires	

setdateyear

Cette fonction utilise comme données d'entrée un horodatage **timestamp** et une année **year** , puis elle met à jour l'horodatage **timestamp** avec l'année **year** spécifiée dans les données d'entrée.

Syntaxe :

```
setdateyear (timestamp, year)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
timestamp	Horodatage Qlik Sense standard (souvent juste une date).
year	Année composée de quatre chiffres.

Exemples et résultats :
Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de script

Exemple	Résultat
`setdateyear ('29/10/2005', 2013)`	Renvoie '29/10/2013.
`setdateyear ('29/10/2005 04:26:14', 2013)`	Renvoie '29/10/2013 04:26:14'. Pour afficher la partie horaire de l'horodatage dans une visualisation, vous devez définir l'option de formatage des nombres sur Date et choisir une valeur de formatage comprenant les valeurs horaires.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante contient les dates initiales et une colonne dans laquelle l'année a été définie sur 2013.

Table des résultats

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

Cette fonction utilise comme données d'entrée un horodatage **timestamp**, un mois **month** et une année **year** , puis elle met à jour l'horodatage **timestamp** avec l'année **year** et le mois **month** spécifiés dans les données d'entrée. .

Syntaxe :

```
SetDateYearMonth (timestamp, year, month)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
timestamp	Horodatage Qlik Sense standard (souvent juste une date).
year	Année composée de quatre chiffres.
month	Mois composé d'un ou de deux chiffres.

Exemples et résultats :
Ces exemples utilisent le format de date **DD/MM/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données. Modifiez le format utilisé dans les exemples en fonction de vos exigences.

Exemples de script

Exemple	Résultat
`setdateyearmonth ('29/10/2005', 2013, 3)`	Renvoie '29/03/2013.
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Renvoie '29/03/2013 04:26:14'. Pour afficher la partie horaire de l'horodatage dans une visualisation, vous devez définir l'option de formatage des nombres sur Date et choisir une valeur de formatage comprenant les valeurs horaires.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante contient les dates initiales et une colonne dans laquelle l'année a été définie sur 2013.

Table des résultats

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

testdates	NewYearMonth
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

Cette fonction renvoie le fuseau horaire tel que défini sur l'ordinateur sur lequel le moteur Qlik est exécuté.

Syntaxe :

```
TimeZone( )
```

Type de données renvoyé : double

Exemple :

```
timezone( )
```

Pour afficher un fuseau horaire différent dans une mesure de votre application, utilisez la fonction `localtime()` dans une mesure.

today

Cette fonction renvoie la date actuelle. La fonction renvoie des valeurs au format de variable système `DateFormat`.

Syntaxe :

```
today([ timer_mode])
```

Type de données renvoyé : double

La fonction `today()` peut être utilisée dans le script de chargement ou dans des objets graphiques.

La valeur `timer_mode` par défaut est 1.

Arguments

Argument	Description
timer_mode	Admet les valeurs suivantes : 0 (jour du dernier chargement de données terminé) 1 (jour de l'appel de la fonction) 2 (jour d'ouverture de l'application) *Si vous utilisez la fonction dans un script de chargement, **timer_mode=0** calcule le jour du dernier chargement de données terminé, tandis que **timer_mode=1** génère le jour du chargement de données actif.*

Exemples de fonction

Valeur de timer_mode	Résultat en cas d'utilisation dans un script de chargement	Résultat en cas d'utilisation dans un objet graphique
0	Renvoie une date, au format de la variable système `DateFormat`, du dernier chargement de données réussi précédant le dernier chargement de données.	Renvoie une date, au format de la variable système `DateFormat`, du dernier chargement de données.
1	Renvoie une date, au format de la variable système `DateFormat`, du dernier chargement de données.	Renvoie une date, au format de la variable système `DateFormat`, de l'appel de la fonction.
2	Renvoie une date, au format de la variable système `DateFormat`, correspondant au début de la session de l'utilisateur dans l'application. Cette valeur ne sera pas mise à jour, sauf si l'utilisateur charge de nouveau le script.	Renvoie la date, au format de la variable système `DateFormat`, correspondant au début de la session de l'utilisateur dans l'application. Cette valeur sera actualisée une fois qu'une nouvelle session commencera ou lorsque les données de l'application seront de nouveau chargées.

Cas d'utilisation

La fonction `today()` est couramment utilisée comme composant d'une expression. Par exemple, elle peut être utilisée pour calculer les intérêts cumulés au cours d'un mois jusqu'à la date du jour.

Le tableau suivant explique le résultat renvoyé par la fonction `today()` suivant différentes valeurs pour l'argument `timer_mode` :

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – génération d'objets via un script de chargement

Script de chargement et résultats

Vue d'ensemble

L'exemple suivant crée trois variables avec la fonction `today()`. Chaque variable utilise l'une des options `timer_mode` pour démontrer leur effet.

Pour que les variables démontrent leur objectif, chargez le script, puis, au bout de 24 heures, chargez-le une deuxième fois. Cela a pour résultat d'afficher des valeurs différentes dans les variables `today(0)` et `today(1)`, ce qui permet de démontrer correctement leur objectif.

Script de chargement

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Résultats

Une fois les données chargées pour la deuxième fois, créez trois zones de texte en suivant les instructions ci-dessous.

Commencez par créer une zone de texte pour les données précédemment chargées.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vPreviousDataLoad`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Previous Reload Time' à l'objet.

Créez ensuite une zone de texte pour les données en cours de chargement.

Procédez comme suit :

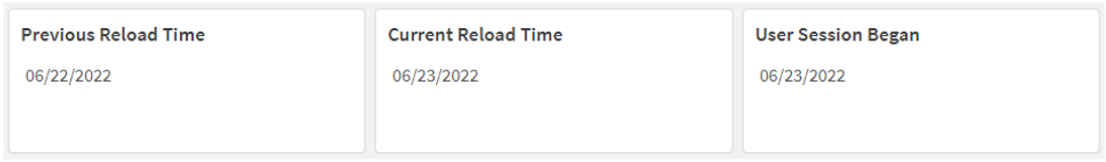
1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vCurrentDataLoad`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Current Reload Time' à l'objet.

Créez une dernière zone de texte à afficher lors du démarrage de la session de l'utilisateur dans l'application.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante à l'objet :
   `=vApplicationOpened`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'User Session Started' à l'objet.

Diagramme de variables créées via la fonction `today()` dans le script de chargement



L'image ci-dessus montre des exemples de valeur pour chacune des variables créées. Par exemple, les valeurs pourraient être comme suit :

- Date du chargement précédent : 06/22/2022
- Date du chargement actuel : 06/23/2022
- Date de début de la session utilisateur : 06/23/2022

Exemple 2 – génération d'objets sans script de chargement

Script de chargement et expression de graphique

Vue d'ensemble

L'exemple suivant crée trois objets graphiques avec la fonction `today()`. Chaque objet graphique utilise l'une des options `timer_mode` pour démontrer leur effet.

Pour cet exemple, il n'existe aucun script de chargement.

Résultats

Une fois les données chargées pour la deuxième fois, créez trois zones de texte.

Commencez par créer une zone de texte pour le dernier chargement de données.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :
   `=today(0)`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Dernier chargement de données' à l'objet.

Créez ensuite une zone de texte affichant l'heure actuelle.

Procédez comme suit :

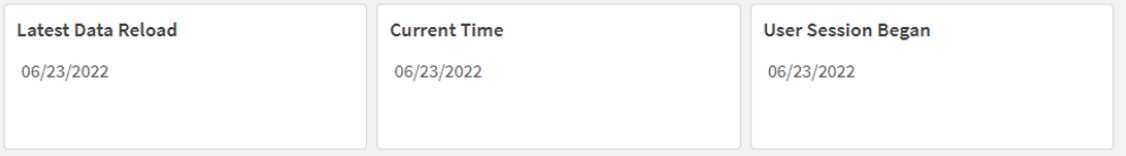
1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :
   `=today(1)`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Heure actuelle' à l'objet.

Créez une dernière zone de texte à afficher lors du démarrage de la session de l'utilisateur dans l'application.

Procédez comme suit :

1. À l'aide de l'objet graphique **Text et image**, créez une zone de texte.
2. Ajoutez la mesure suivante :
   `=today(2)`
3. Sous **Aspect**, sélectionnez **Show titles** et ajoutez le titre 'Début de la session utilisateur' à l'objet.

Diagramme d'objets créés via la fonction `today()` sans script de chargement



L'image ci-dessus montre des exemples de valeur pour chacun des objets créés. Par exemple, les valeurs pourraient être comme suit :

- Dernier chargement de données : 06/23/2022
- Date actuelle : 06/23/2022
- Date de début de la session utilisateur : 06/23/2022

L'objet graphique 'Dernier chargement de données' utilise une valeur `timer_mode` égale à 0. Cela renvoie l'horodatage du dernier chargement de données réussi.

L'objet graphique 'Date actuelle' utilise une valeur `timer_mode` égale à 1. Cela renvoie la date actuelle conformément à l'horloge système. Si la feuille ou l'objet est actualisé, cette valeur sera mise à jour.

L'objet graphique 'Début de la session utilisateur' utilise une valeur `timer_mode` égale à 2. Cela renvoie l'horodatage de l'ouverture de l'application et du début de la session utilisateur.

Exemple 3 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de soldes de prêts, chargé dans une table appelée `Loans`.
- Données de la table avec des champs pour l'ID de prêt, le solde au début du mois et le taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels qui ont été accumulés pour chaque prêt au cours du mois jusqu'à la date du jour. Même si l'application est actualisée une seule fois par semaine, l'utilisateur souhaite que les résultats soient actualisés chaque fois que l'objet ou l'application est actualisé(e).

Script de chargement

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Ajoutez les champs suivants comme dimensions :

- loan_id
- start_balance

3. Ensuite, pour calculer les intérêts accumulés, créez une mesure : `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

La fonction `monthstart()`, avec la fonction `today()` utilisée pour renvoyer la date d'aujourd'hui comme seul argument, renvoie la date de début du mois en cours. Si on soustraie ce résultat de la date actuelle, de nouveau avec la fonction `today()`, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent ce mois.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif encouru pour cette période. Le résultat est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts cumulés jusqu'à présent ce mois.

Étant donné que la valeur 1 est utilisée comme argument `timer_mode` dans les fonctions `today()` au sein de l'expression, chaque fois que l'objet graphique est actualisé (via l'ouverture de l'application, l'actualisation de la page, la navigation entre les feuilles, etc.), la date renvoyée est la date actuelle et les résultats sont actualisés en conséquence.

UTC

Renvoie la valeur actuelle de l'argument Coordinated Universal Time.

Syntaxe :

```
UTC( )
```

Type de données renvoyé : double

Exemple :

```
utc( )
```

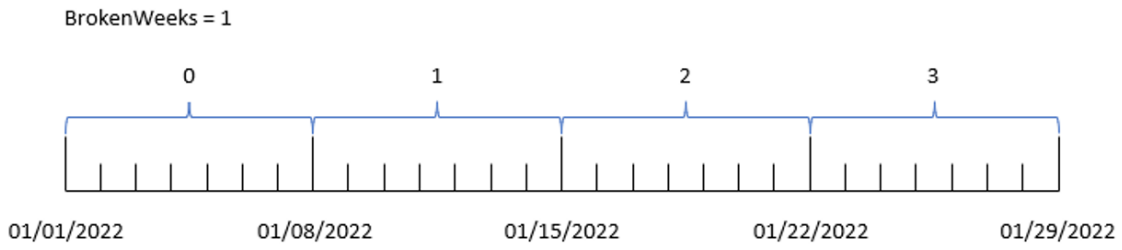
week

Cette fonction renvoie un entier représentant le numéro de la semaine selon la norme ISO 8601. Le numéro de la semaine est calculé à partir de l'interprétation de date de l'expression, conformément à l'interprétation standard des nombres.

Syntaxe :

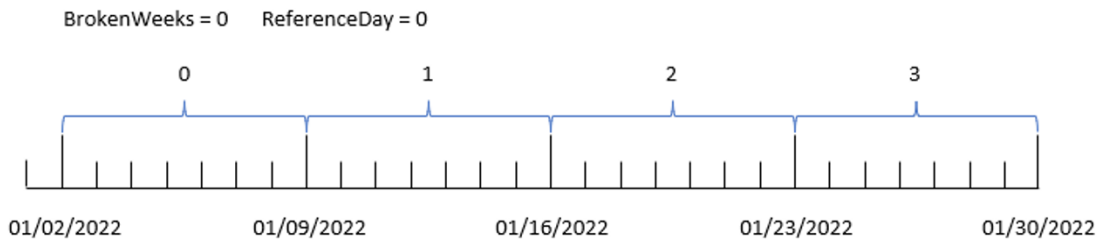
```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Exemple de diagramme de la fonction week(), avec des semaines interrompues



Le comptage des numéros de semaine commence le 1er janvier (car Qlik Sense est défini par défaut pour utiliser des semaines interrompues). La première semaine se termine le jour précédant la variable système `FirstWeekDay`, quel que soit le nombre de jours de cette semaine-là. La variable système `FirstWeekDay` peut être remplacée au sein de la fonction `week()` par l'argument `first_week_day`.

Exemple de diagramme de la fonction week(), avec des semaines ininterrompues et ReferenceDay=0



La fonction `week()` permet aussi de spécifier s'il faut utiliser des semaines interrompues ou ininterrompues via l'argument `broken_weeks`. Si la fonctionnalité de semaines interrompues est employée, la semaine 1 doit contenir un certain nombre de jours en janvier, tel que défini par la variable système `ReferenceDay`. Par conséquent, la semaine 1 peut potentiellement commencer en décembre, ou, sinon, la semaine 52 ou 53 peut se poursuivre en janvier. Pour finir, l'argument `reference_day` permet à la fonction de remplacer la variable système `ReferenceDay`.

Contrairement à la fonction `weekname()`, la fonction `week()` ne renvoie pas également la valeur year. Cela permet aux agrégations de comparer des semaines sur des années.

Dans cette fonction, quatre arguments peuvent être utilisés.

Argument n° 1 : timestamp

Il s'agit de la date à évaluer sous forme d'horodatage ou d'expression aboutissant à un horodatage, pour convertir, par exemple, '2012-10-12'.

Argument n° 2 : first_week_day

Si vous ne précisez pas la variable **first_week_day**, la valeur de la variable **FirstWeekDay** sera utilisée comme premier jour de la semaine.

Si vous souhaitez utiliser un autre jour comme premier jour de la semaine, définissez les valeurs suivantes pour la variable **first_week_day** :

- 0 pour lundi
- 1 pour mardi
- 2 pour mercredi
- 3 pour jeudi
- 4 pour vendredi
- 5 pour samedi
- 6 pour dimanche

L'entier renvoyé par la fonction utilise désormais le premier jour de la semaine que vous avez défini grâce à la variable **first_week_day**.

Argument n° 3 : broken_weeks

Si vous ne précisez pas la variable **broken_weeks**, la valeur de la variable **BrokenWeeks** sera utilisée pour définir si les semaines sont interrompues ou non.

Par défaut, les fonctions Qlik Sense utilisent des semaines ininterrompues. Autrement dit :

- Certaines années, la semaine 1 commence en décembre, tandis que d'autres années, la semaine 52 ou 53 se poursuit en janvier.
- La semaine 1 comporte toujours au moins 4 jours en janvier.

L'alternative consiste à utiliser des semaines interrompues.

- La semaine 52 ou 53 ne se poursuit pas en janvier.
- La semaine 1 débute le 1er janvier et n'est, dans la plupart des cas, pas une semaine complète.

Les valeurs suivantes peuvent être utilisées :

- 0 (= utilisation de semaines ininterrompues)
- 1 (= utilisation de semaines interrompues)

Argument n° 4 : reference_day

Si vous ne spécifiez pas **reference_day**, la valeur de la variable **ReferenceDay** sera utilisée pour spécifier le jour du mois de janvier devant être défini comme jour de référence pour définir la semaine 1. Par défaut, les fonctions Qlik Sense utilisent le 4 comme jour de référence. Cela signifie que la semaine 1 doit contenir le 4 janvier ou, en d'autres termes, que la semaine 1 doit toujours comprendre au moins 4 jours en janvier.

Les valeurs suivantes peuvent s'utiliser pour définir un autre jour de référence :

- 1 (= 1er janvier)
- 2 (= 2 janvier)
- 3 (= 3 janvier)
- 4 (= 4 janvier)
- 5 (= 5 janvier)
- 6 (= 6 janvier)
- 7 (= 7 janvier)

Cas d'utilisation

La fonction `The week()` est utile lorsque vous souhaitez comparer des agrégations par semaine. Par exemple, elle peut être utilisée si vous souhaitez voir les ventes totales de produits par semaine. La fonction `week()` est préférée à la fonction `weekname()` lorsque l'utilisateur souhaite que le calcul n'utilise pas forcément la variable système `BrokenWeeks`, `FirstWeekDay` ou `ReferenceDay` de l'application.

De plus, la fonction `week()` est celle sélectionnée lorsque vous souhaitez effectuer une comparaison sur plusieurs années. Grâce à la fonction `week()`, l'utilisateur peut créer sa propre combinaison de ces variables à utiliser dans les cas dans lesquels la fonction est utilisée.

Ces dimensions peuvent être créées soit dans le script de chargement, en utilisant la fonction pour créer un champ dans une table Master Calendar, soit directement dans un graphique sous forme de dimension calculée.

Exemples de fonction

Exemple	Résultat
`week( '10/12/2012')`	Renvoie 41.
`week( '35648')`	Renvoie 32, car 35648 = 08/06/1997.
`week('10/12/2012', 0, 1)`	Renvoie 42.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – variables système par défaut

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2021 et les deux premières semaines de 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `week_number`, qui renvoie l'année et le numéro de semaine des transactions.
- Création d'un champ appelé `week_day`, montrant la valeur weekday de chaque date de transaction.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
```

```
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `week_day`
- `week_number`

Tableau de résultats

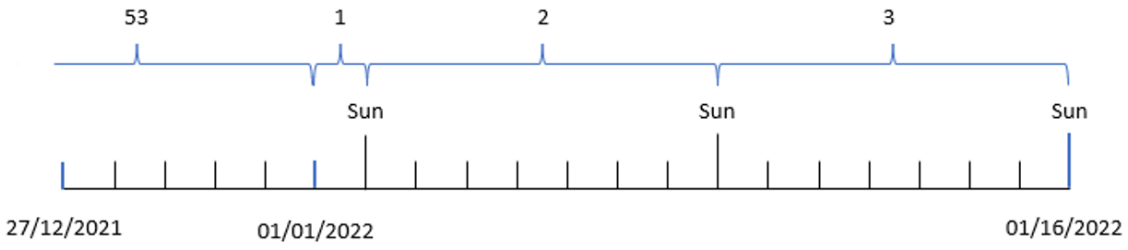
id	date	week_day	week_number
8183	12/27/2021	Mon	53
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	2
8190	01/03/2022	Mon	2
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	3
8197	01/10/2022	Mon	3
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Le champ `week_number` est créé dans l'instruction LOAD précédente via la fonction `week()` et en transmettant le champ `date` comme argument de la fonction.

Aucun autre paramètre n'est transmis à la fonction. Par conséquent, les variables par défaut qui affectent la fonction `week()` sont les suivantes :

- `BrokenWeeks` : Le comptage des semaines commence le 1er janvier.
- `FirstWeekDay` : Le premier jour de la semaine est un dimanche.

Diagramme de la fonction `week()`, utilisant les variables système par défaut



Étant donné que l'application utilise la variable système `BrokenWeeks` par défaut, la semaine 1 commence le 1er janvier, un samedi.

En raison de la variable système `FirstWeekDay` par défaut, les semaines commencent un dimanche. Le premier dimanche après le 1er janvier est le 2 janvier, date à laquelle commence la semaine 2.

Exemple 2 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Création d'un champ, `week_number`, qui renvoie l'année et le numéro de semaine des transactions.
- Création d'un champ appelé `week_day`, montrant la valeur weekday de chaque date de transaction.

Dans cet exemple, nous voulons déterminer le début de la semaine de travail sur le mardi.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
```

```
        Week(date,1) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- week_day
- week_number

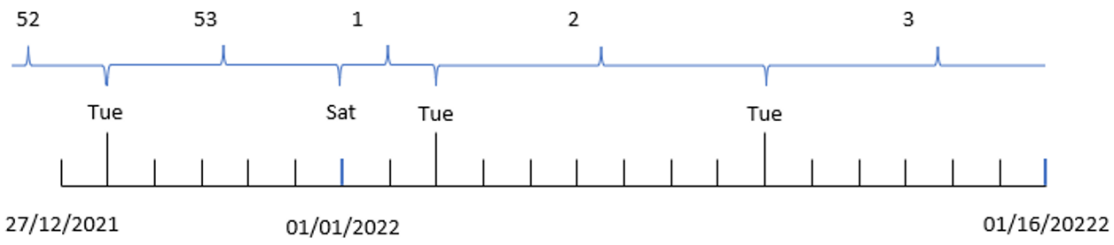
Tableau de résultats

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1

id	date	week_day	week_number
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

L'application continue à utiliser des semaines interrompues. Cependant, l'argument `first_week_day` a été défini sur 1 dans la fonction `week()`. Cela détermine le premier jour de la semaine comme étant un mardi.

Diagramme de la fonction `week()`, exemple first_week_day



L'application utilise la variable système `BrokenWeeks` par défaut ; par conséquent, la semaine 1 commence le 1er janvier, un samedi.

L'argument `first_week_day` de la fonction `week()` détermine le premier jour de la semaine comme étant un mardi. Par conséquent, la semaine 53 commence le 28 décembre 2021.

Cependant, étant donné que la fonction continue à utiliser des semaines interrompues, la semaine 1 ne dure que deux jours, car le premier mardi après le 1er janvier tombe le 3 janvier.

Exemple 3 – unbroken_weeks

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Dans cet exemple, nous utilisons des semaines ininterrompues.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,6,0) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `week_day`
- `week_number`

Diagramme de la fonction `week()`, exemple objet graphique

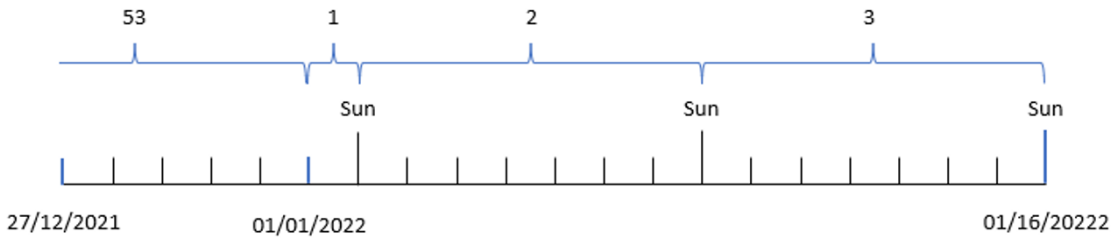


Tableau de résultats

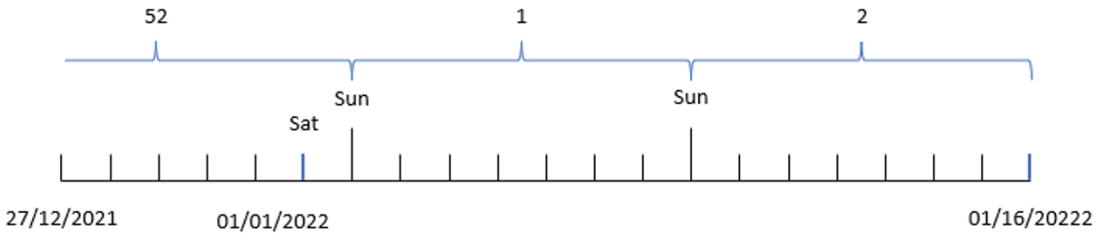
id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	52
8185	12/29/2021	Wed	52
8186	12/30/2021	Thu	52
8187	12/31/2021	Fri	52
8188	01/01/2022	Sat	52
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	1
8192	01/05/2022	Wed	1
8193	01/06/2022	Thu	1
8194	01/07/2022	Fri	1
8195	01/08/2022	Sat	1
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	2

id	date	week_day	week_number
8199	01/12/2022	Wed	2
8200	01/13/2022	Thu	2
8201	01/14/2022	Fri	2

Le paramètre first_week_date est défini sur 1, faisant du mardi le premier jour de la semaine. Le paramètre broken_weeks est défini sur 0, obligeant la fonction à utiliser des semaines ininterrompues. Pour finir, le troisième paramètre définit la valeur reference_day sur 2.

Le paramètre `first_week_date` est défini sur 6, faisant du dimanche le premier jour de la semaine. Le paramètre `broken_weeks` est défini sur 0, obligeant la fonction à utiliser des semaines ininterrompues.

Diagramme de la fonction week(), exemple utilisant des semaines ininterrompues



Si on utilise des semaines ininterrompues, la semaine 1 ne commence pas forcément le 1er janvier ; en revanche, elle doit compter au moins quatre jours. Par conséquent, dans l'ensemble de données, la semaine 52 se termine le samedi 1er janvier 2022. La semaine 1 commence donc le jour de la variable système `FirstWeekDay`, à savoir, le dimanche 2 janvier. Cette semaine se termine le samedi suivant, le 8 janvier.

Exemple 4 – reference_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du troisième exemple.
- Création d'un champ, `week_number`, qui renvoie l'année et le numéro de semaine des transactions.
- Création d'un champ appelé `week_day`, montrant la valeur weekday de chaque date de transaction.

En outre, les conditions suivantes doivent être remplies :

- La semaine de travail commence un mardi.
- L'entreprise utilise des semaines ininterrompues.
- La valeur `reference_day` est égale à 2. En d'autres termes, le nombre minimal de jours en janvier la semaine 1 est de 2.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1,0,2) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`

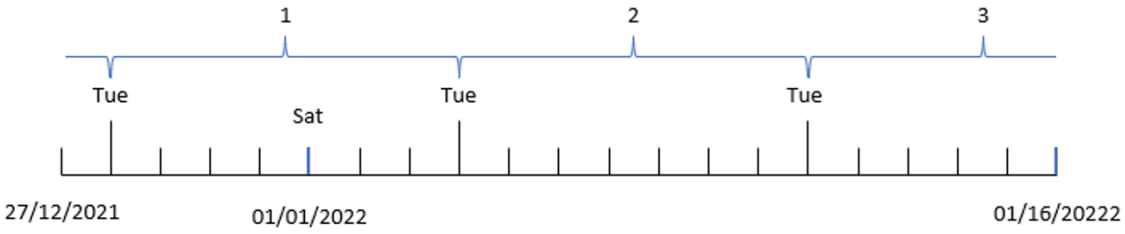
- week_day
- week_number

Tableau de résultats

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	1
8185	12/29/2021	Wed	1
8186	12/30/2021	Thu	1
8187	12/31/2021	Fri	1
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Le paramètre `first_week_date` est défini sur 1, faisant du mardi le premier jour de la semaine. Le paramètre `broken_weeks` est défini sur 0, obligeant la fonction à utiliser des semaines ininterrompues. Pour finir, le troisième paramètre définit le paramètre `reference_day` sur 2.

Diagramme de la fonction week(), exemple reference_day



Étant donné que la fonction utilise des semaines ininterrompues et qu'une valeur `reference_day` égale à 2 est utilisée comme paramètre, la semaine 1 n'a besoin d'inclure que deux jours en janvier. Le premier jour de la semaine étant un mardi, la semaine 1 commence le 28 décembre 2021 et se termine le lundi 3 janvier 2022.

Exemple 5 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie le numéro de semaine est créé sous forme de mesure dans un objet graphique.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
```

```
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Ajoutez les champs suivants comme dimensions :
   - `id`
   - `date`
3. Ensuite, créez la mesure suivante :
   `=week (date)`
4. Créez une mesure , `week_day` pour indiquer la valeur weekday de chaque date de transaction :
   `=weekday(date)`

Tableau de résultats

id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Mon
8184	12/28/2021	53	Tue
8185	12/29/2021	53	Wed
8186	12/30/2021	53	Thu
8187	12/31/2021	53	Fri
8188	01/01/2022	1	Sat
8189	01/02/2022	2	Sun
8190	01/03/2022	2	Mon
8191	01/04/2022	2	Tue
8192	01/05/2022	2	Wed
8193	01/06/2022	2	Thu
8194	01/07/2022	2	Fri
8195	01/08/2022	2	Sat
8196	01/09/2022	3	Sun
8197	01/10/2022	3	Mon

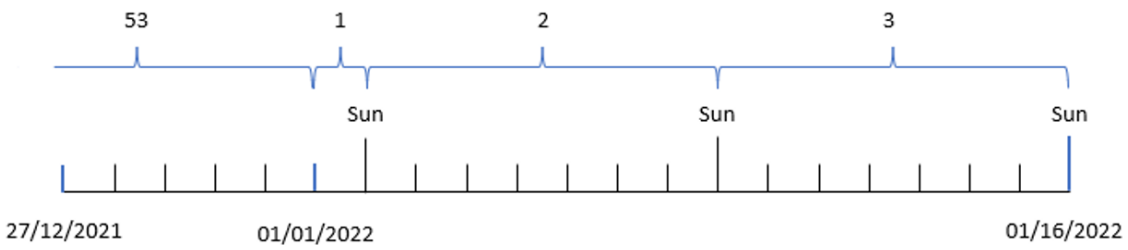
id	date	=week(date)	=weekday(date)
8198	01/11/2022	3	Tue
8199	01/12/2022	3	Wed
8200	01/13/2022	3	Thu
8201	01/14/2022	3	Fri

Le champ `week_number` est créé dans l'instruction LOAD précédente via la fonction `week()` et en transmettant le champ `date` comme argument de la fonction.

Aucun autre paramètre n'est transmis à la fonction. Par conséquent, les variables par défaut qui affectent la fonction `week()` sont les suivantes :

- `BrokenWeeks` : Le comptage des semaines commence le 1er janvier.
- `FirstWeekDay` : Le premier jour de la semaine est un dimanche.

Diagramme de la fonction `week()`, exemple objet graphique



Étant donné que l'application utilise la variable système `BrokenWeeks` par défaut, la semaine 1 commence le 1er janvier, un samedi.

En raison de la variable système `FirstWeekDay` par défaut, les semaines commencent un dimanche. Le premier dimanche après le 1er janvier est le 2 janvier, date à laquelle commence la semaine 2.

Exemple 6 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2019 et les deux premières semaines de 2020, chargé dans une table appelée `Transactions`.

- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

L'application utilise principalement des semaines interrompues sur son tableau de bord. Cependant, l'utilisateur final souhaite un objet graphique présentant les ventes totales par semaine via des semaines ininterrompues. Le jour de référence doit être le 2 janvier, avec des semaines commençant un mardi. Cela est possible même lorsque cette dimension n'est pas disponible dans le modèle de données, en utilisant la fonction `week()` comme dimension calculée dans le graphique.

Script de chargement

```
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez un tableau.
2. Créez la dimension calculée suivante :
   `=week(date)`
3. Ensuite, créez la mesure d'agrégation suivante :
   `=sum(amount)`
4. Définissez le **Formatage des nombres** des mesures sur **Devise**.

5. Sélectionnez le menu **Tri** et, pour la dimension calculée, supprimez le tri personnalisé.
6. Désélectionnez les options **Trier par nombre** et **Trier par ordre alphabétique**.

Tableau de résultats

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Cette fonction renvoie une valeur double avec :

- Un nom de jour tel que défini dans la variable d'environnement **DayNames**.
- Un entier compris entre 0 et 6 correspondant au jour nominal de la semaine (0-6).

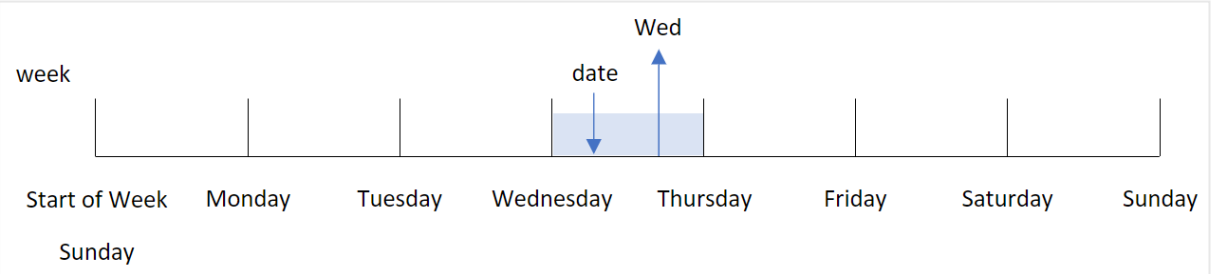
Syntaxe :

```
weekday(date [,first_week_day=0])
```

Type de données renvoyé : double

La fonction `weekday()` détermine le jour de la semaine d'une date. Elle renvoie ensuite une valeur de chaîne représentant ce jour-là.

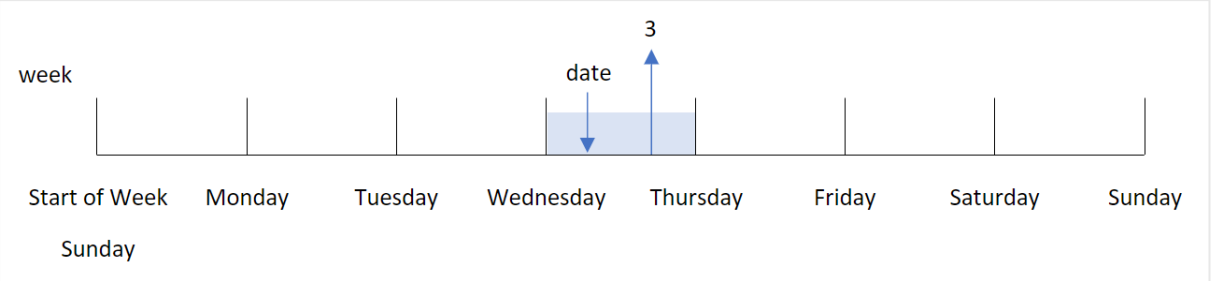
Diagramme de la fonction `weekday()` renvoyant le nom du jour d'une date



Le résultat renvoie la valeur numérique correspondant à ce jour de la semaine (0-6), en fonction du jour de début de la semaine. Par exemple, si le premier jour de la semaine est défini comme étant un dimanche, un mercredi renverra une valeur numérique `3`. Ce jour de début est déterminé par la variable système `FirstWeekDay` ou par le paramètre de fonction `first_week_day`.

Vous pouvez utiliser cette valeur numérique dans le cadre d'une expression arithmétique. Par exemple, vous pouvez la multiplier par `1` pour renvoyer la valeur elle-même.

Diagramme de la fonction weekday() avec la valeur numérique du jour affichée au lieu du nom du jour



Cas d'utilisation

La fonction `weekday()` est utile lorsque vous souhaitez comparer des agrégations par jour de la semaine. Par exemple, si vous souhaitez comparer les ventes moyennes de produits par jour de la semaine.

Ces dimensions peuvent être créées soit dans le script de chargement, en utilisant la fonction pour créer un champ dans une table **Master Calendar**, soit directement dans un graphique sous forme de mesure calculée.

Rubriques connexes

Rubriques	Interaction
FirstWeekDay (page 221)	Définit le jour de début de chaque semaine.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
first_week_day	Spécifie le jour où débute la semaine. S'il est omis, c'est la valeur de la variable **FirstWeekDay** qui est utilisée. *FirstWeekDay (page 221)*

Vous pouvez utiliser les valeurs suivantes pour définir le premier jour de la semaine dans l'argument `first_week_day` :

first_week_day values

Jour	Valeur
Lundi	0
Mardi	1
Mercredi	2
Jeudi	3
Vendredi	4
Samedi	5
Dimanche	6

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Sauf indication contraire, la valeur `FirstWeekDay` est définie sur `0` dans ces exemples.

Exemples de fonction

Exemple	Résultat
`weekday('10/12/1971')`	Renvoie 'Tue' (mardi) et 1.
`weekday('10/12/1971' , 6)`	Renvoie 'Tue' (mardi) et 2. Dans cet exemple, le dimanche (`6`) est de premier jour de la semaine.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Renvoie 'Tue' (mardi) et 2.

Exemple 1 - chaîne Weekday

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur 6 (dimanche).
- Variable `DayNames` définie de sorte à utiliser les noms de jour par défaut.

- Chargement précédent contenant la fonction `weekday()`, définie comme le champ '`week_day`' et qui renvoie le jour de la semaine des transactions.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `week_day`

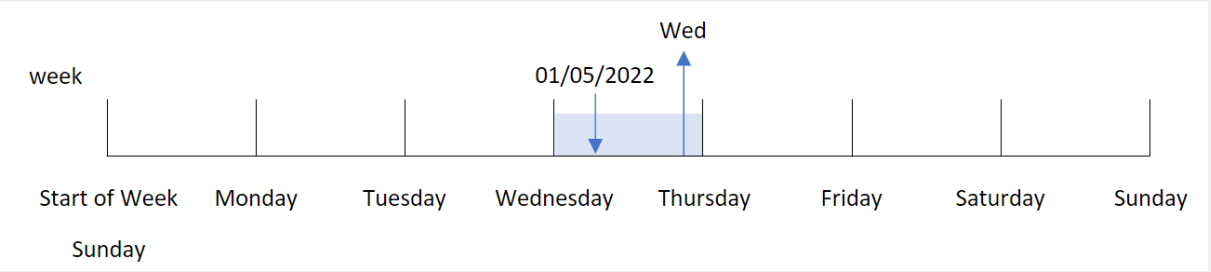
Tableau de résultats

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Le champ 'week_day' est créé dans l'instruction LOAD précédente via la fonction weekday() et en transmettant le champ date comme argument de la fonction.

La fonction weekday() renvoie la valeur de chaîne weekday, à savoir, le nom du jour de la semaine défini par la variable système DayNames.

Diagramme de la fonction weekday() renvoyant le mercredi comme jour de la semaine pour la transaction 8192



La transaction 8192 a eu lieu le 5 janvier. La variable système FirstWeekDay définit le premier jour de la semaine comme étant le dimanche. La transaction de la fonction weekday() a eu lieu un mercredi et renvoie cette valeur, au format abrégé de la variable système DayNames, dans le champ week_day.

Les valeurs du champ 'week_day' sont alignées à droite dans la colonne, en raison d'un résultat double contenant du texte et un chiffre pour le champ (Wednesday, 3). Pour convertir la valeur du champ en équivalent numérique, il est possible d'intégrer le champ à la fonction num(). Par exemple, dans la transaction 8192, la valeur Wednesday (mercredi) serait convertie en chiffre 3.

Exemple 2 - first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée 'Transactions'.
- Variable système FirstWeekDay définie sur 6 (dimanche).
- Variable DayNames définie de sorte à utiliser les noms de jour par défaut.
- Chargement précédent contenant la fonction weekday(), définie comme le champ 'week_day' et qui renvoie le jour de la semaine des transactions.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
```

```
    Load
       *,
       WeekDay(date,1) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Résultats

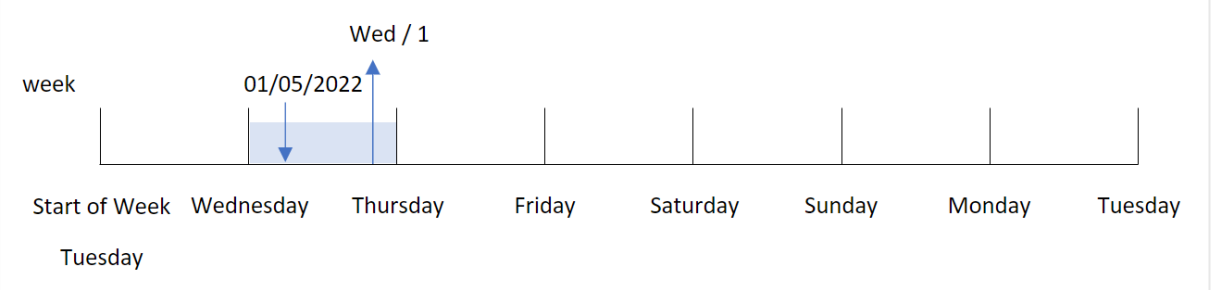
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- week_day

Tableau de résultats

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Diagramme de la fonction `weekday()` montrant que le mercredi porte la valeur numérique double 1



Étant donné que l'argument `first_week_day` égal à `1` est utilisé dans la fonction `weekday()`, le premier jour de la semaine est le mardi. Par conséquent, toutes les transactions qui ont lieu un mardi auront une valeur numérique double `0`.

La transaction 8192 a eu lieu le 5 janvier. La fonction `weekday()` identifie qu'il s'agit un mercredi ; par conséquent, l'expression renvoie la valeur numérique double `1`.

Exemple 3 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur `6` (dimanche).
- Variable `DayNames` définie de sorte à utiliser les noms de jour par défaut.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui identifie la valeur weekday est créé sous forme de mesure dans un graphique de l'application.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date

Pour calculer la valeur weekday, créez la mesure suivante :

- =weekday(date)

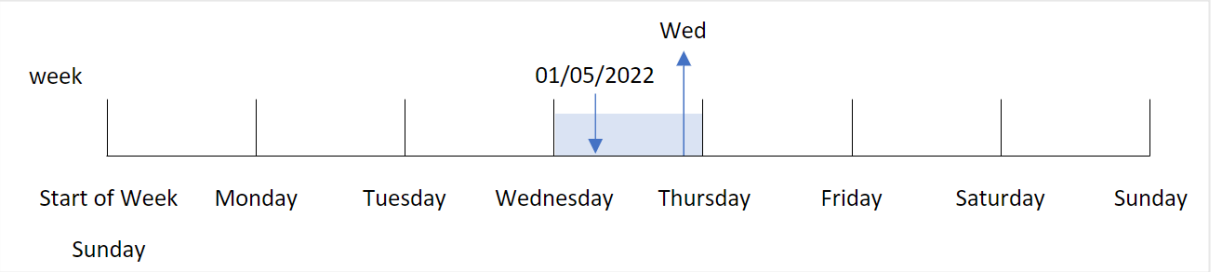
Tableau de résultats

id	date	=weekday(date)
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Le champ '=weekday(date)' est créé dans le graphique via la fonction weekday() et en transmettant le champ date comme argument de la fonction.

La fonction weekday() renvoie la valeur de chaîne weekday, à savoir, le nom du jour de la semaine défini par la variable système DayNames.

Diagramme de la fonction weekday() renvoyant le mercredi comme jour de la semaine pour la transaction 8192



La transaction 8192 a eu lieu le 5 janvier. La variable système FirstWeekDay définit le premier jour de la semaine comme étant le dimanche. La transaction de la fonction weekday() a eu lieu un mercredi et renvoie cette valeur, au format abrégé de la variable système DayNames, dans le champ =weekday(date).

Exemple 4 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `FirstWeekDay` définie sur `6` (dimanche).
- Variable `DayNames` définie de sorte à utiliser les noms de jour par défaut.

L'utilisateur final souhaite un graphique présentant les ventes moyennes par jour de la semaine pour les transactions.

Script de chargement

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
LOAD
  RecNo() AS id,
  MakeDate(2022, 1, Ceil(Rand() * 31)) as date,
  Rand() * 1000 AS amount

Autogenerate(1000);
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `=weekday(date)`
- `=avg(amount)`

Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

weekday(date)	Avg(amount)
Sun	$536.96
Mon	$500.80
Tue	$515.63
Wed	$509.21
Thu	$482.70
Fri	$441.33
Sat	$505.22

weekend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour (dimanche) de la semaine calendaire contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
WeekEnd(date [, period_no[, first_week_day]])
```

Type de données renvoyé : double

Exemple de diagramme de la fonction weekend()

La fonction `weekend()` détermine la semaine au cours de laquelle tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la dernière milliseconde de cette semaine-là. Le premier jour de la semaine est déterminé par la variable d'environnement `FirstWeekDay`. Cependant, cela peut être remplacé par l'argument `first_week_day` dans la fonction `weekend()`.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**shift** est un entier, où la valeur 0 indique la semaine comprenant l'argument **date**. Les valeurs négatives spécifiées pour shift indiquent les semaines passées tandis que les valeurs positives désignent les semaines à venir.
first_week_day	Spécifie le jour où débute la semaine. S'il est omis, c'est la valeur de la variable **FirstWeekDay** qui est utilisée. Les valeurs possibles de **first_week_day** sont 0 pour lundi, 1 pour mardi, 2 pour mercredi, 3 pour jeudi, 4 pour vendredi, 5 pour samedi et 6 pour dimanche. Pour plus d'informations sur la variable système, voir *FirstWeekDay (page 221)*.
broken_weeks	Si vous ne précisez pas la variable **broken_weeks**, la valeur de la variable **BrokenWeeks** sera utilisée pour définir si les semaines sont interrompues ou non.

Cas d'utilisation

La fonction `weekend()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise les jours restants de la semaine pour la date spécifiée. Par exemple, elle peut être utilisée si un utilisateur souhaite calculer le total des intérêts non encore encourus au cours de la semaine.

Exemple	Résultat
`weekend('01/10/2013')`	Renvoie 01/12/2013 23:59:59.
`weekend('01/10/2013', -1)`	Renvoie 01/05/2013 23:59:59..
`weekend('01/10/2013', 0, 1)`	Renvoie 01/14/2013 23:59:59.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `end_of_week`, qui renvoie un horodatage de la fin de la semaine au cours de laquelle les transactions ont eu lieu.

Script de chargement

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekend(date) as end_of_week,
        timestamp(weekend(date)) as end_of_week_timestamp
```

```
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tableau de résultats

date	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Le champ `end_of_week` est créé dans l'instruction LOAD précédente via la fonction `weekend()` et en transmettant le champ date comme argument de la fonction.

La fonction `weekend()` identifie la semaine de la valeur date et renvoie un horodatage pour la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `weekend()`, exemple de base

La transaction 8191 a eu lieu le 5 février. La variable système `FirstWeekDay` définit le premier jour de la semaine comme étant un dimanche. La fonction `weekend()` identifie que le premier samedi après le 5 février – et par conséquent la fin de la semaine – était le 5 février. Par conséquent, la valeur `end_of_week` pour cette transaction renvoie la dernière milliseconde de ce jour-là, le 5 février à 11:59:59 PM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_week_end`, qui renvoie l'horodatage du début de la semaine avant la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        weekend(date,-1) as previous_week_end,
        timestamp(weekend(date,-1)) as previous_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

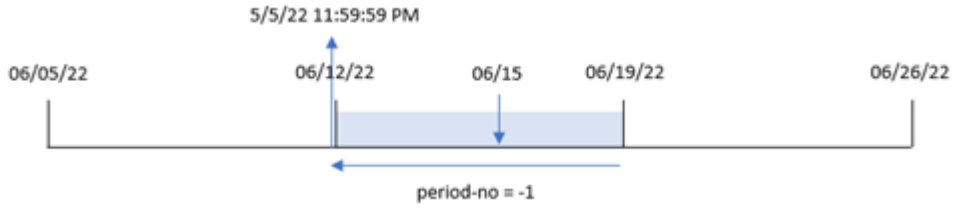
- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

Tableau de résultats

date	end_of_week	end_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 11:59:59 PM
1/19/2022	01/15/2022	1/15/2022 11:59:59 PM
2/5/2022	01/29/2022	1/29/2022 11:59:59 PM
2/28/2022	02/26/2022	2/26/2022 11:59:59 PM
3/16/2022	03/12/2022	3/12/2022 11:59:59 PM
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM
9/26/2022	09/24/2022	9/24/2022 11:59:59 PM
10/14/2022	10/08/2022	10/8/2022 11:59:59 PM
10/29/2022	10/22/2022	10/22/2022 11:59:59 PM

Dans cet exemple, étant donné qu'un argument `period_no` égal à -1 a été utilisé comme argument de décalage dans la fonction `weekend()`, la fonction commence par identifier la semaine au cours de laquelle les transactions ont lieu. Elle regarde ensuite une semaine avant et identifie la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `weekend()`, exemple period_no



La transaction 8196 a eu lieu le 15 juin. La fonction `weekend()` identifie que la semaine commence le 12 juin. Par conséquent, la semaine précédente se termine le 11 juin à 11:59:59 PM ; il s'agit de la valeur renvoyée pour le champ `previous_week_end`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, nous devons définir le mardi comme le premier jour de la semaine de travail.

Script de chargement

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

Transactions:
    Load
        *,
        weekend(date,0,1) as end_of_week,
        timestamp(weekend(date,0,1)) as end_of_week_timestamp,
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

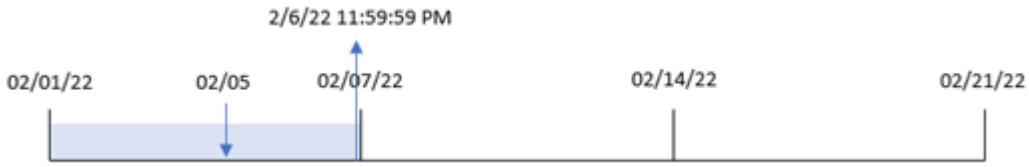
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tableau de résultats

date	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	5/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	5/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM
7/22/2022	07/25/2022	7/25/2022 11:59:59 PM
7/23/2022	07/25/2022	7/25/2022 11:59:59 PM
7/27/2022	08/01/2022	8/1/2022 11:59:59 PM
8/2/2022	08/08/2022	8/8/2022 11:59:59 PM
8/8/2022	08/08/2022	8/8/2022 11:59:59 PM
8/19/2022	08/22/2022	8/22/2022 11:59:59 PM
9/26/2022	09/26/2022	9/26/2022 11:59:59 PM
10/14/2022	10/17/2022	10/17/2022 11:59:59 PM
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Dans ce cas, étant donné que l'argument `first_week_date` égal à 1 est utilisé dans la fonction `weekend()`, il définit le premier jour de la semaine comme étant le mardi.

Diagramme de la fonction weekend(), exemple first_week_day



La transaction 8191 a eu lieu le 5 février. La fonction weekend() identifie que le premier lundi après cette date – et par conséquent la fin de la semaine et la valeur renvoyée – était le 6 février à 11:59:59 PM.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant à la fin de la semaine de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
```

```
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer le début de la semaine d'une transaction, ajoutez les mesures suivantes :

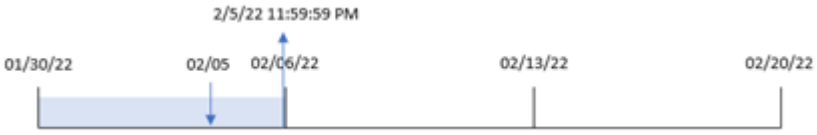
- `=weekend(date)`
- `=timestamp(weekend(date))`

Tableau de résultats

date	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

La mesure `end_of_week` est créée dans l'objet graphique via la fonction `weekend()` et en transmettant le champ date comme argument de la fonction. La fonction `weekend()` identifie la semaine de la valeur date, renvoyant un horodatage pour la dernière milliseconde de cette semaine-là.

Diagramme de la fonction `weekend()`, exemple objet graphique



La transaction 8191 a eu lieu le 5 février. La variable système `FirstWeekDay` définit le premier jour de la semaine comme étant un dimanche. La fonction `weekend()` identifie que le premier samedi après le 5 février – et par conséquent la fin de la semaine – était le 5 février. Par conséquent, la valeur `end_of_week` pour cette transaction renvoie la dernière milliseconde de ce jour-là, le 5 février à 11:59:59 PM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Employee_Expenses`.
- Données constituées des ID des employés, des noms des employés et des notes de frais quotidiennes moyennes de chaque employé.

L'utilisateur final souhaite un objet graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées qu'il reste à encourir pour le reste de la semaine.

Script de chargement

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :
   - `employee_id`
   - `employee_name`
2. Ensuite, pour calculer les intérêts accumulés, créez une mesure : `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

La fonction `weekend()`, en utilisant la date d'aujourd'hui comme son seul argument, renvoie la date de fin de la semaine en cours. Ensuite, en soustrayant la date d'aujourd'hui de la date de fin de la semaine, l'expression renvoie le nombre de jours restants pour cette semaine.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes par employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours du reste de la semaine.

weekname

Cette fonction renvoie une valeur affichant l'année et le numéro de la semaine avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour de la semaine contenant l'argument **date**.

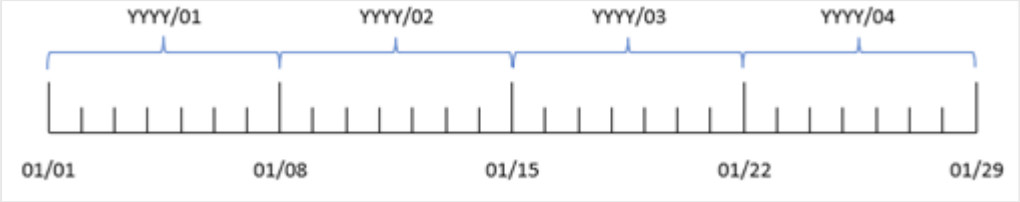
Syntaxe :

```
WeekName(date[, period_no[,first_week_day]])
```

La fonction `weekname()` détermine la semaine dans laquelle la date tombe et renvoie le numéro de semaine et l'année de cette semaine. Le premier jour de la semaine est déterminé par la variable système `FirstWeekDay`. Cependant, vous pouvez également modifier le premier jour de la semaine en utilisant l'argument `first_week_day` dans la fonction `weekname()`.

Par défaut, les applications Qlik Sense utilisent des semaines interrompues (définies par la variable système `BrokenWeeks`) et, par conséquent, le décompte des numéros de semaine commence le 1er janvier et se termine le jour précédant la variable système `FirstWeekDay`, quel que soit le nombre de jours écoulés.

Diagramme de la fonction `weekname`.



Cependant, si votre application utilise des semaines ininterrompues, la semaine 1 peut commencer l'année précédente ou les premiers jours de janvier. Cela dépend de la façon dont vous utilisez les variables système `ReferenceDay` et `FirstWeekDay`.

Cas d'utilisation

La fonction `weekname()` est utile lorsque vous souhaitez comparer des agrégations par semaines.

Par exemple, si vous souhaitez voir les ventes totales de produits par semaine. Pour maintenir la cohérence avec la variable d'environnement `BrokenWeeks` dans l'application, utilisez `weekname()` au lieu de `lunarweekname()`. Si l'application utilise des semaines ininterrompues, la semaine 1 peut contenir des dates de décembre de l'année précédente ou exclure des dates de janvier de l'année en cours. Si l'application utilise des semaines interrompues, la semaine 1 peut contenir moins de sept jours.

Type de données renvoyé : double

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**shift** est un entier, où la valeur 0 indique la semaine comprenant l'argument **date**. Les valeurs négatives spécifiées pour shift indiquent les semaines passées tandis que les valeurs positives désignent les semaines à venir.
first_week_day	Spécifie le jour où débute la semaine. S'il est omis, c'est la valeur de la variable **FirstWeekDay** qui est utilisée. Les valeurs possibles de **first_week_day** sont 0 pour lundi, 1 pour mardi, 2 pour mercredi, 3 pour jeudi, 4 pour vendredi, 5 pour samedi et 6 pour dimanche. Pour plus d'informations sur la variable système, voir *FirstWeekDay (page 221)*.

Vous pouvez utiliser les valeurs suivantes pour définir le premier jour de la semaine dans l'argument `first_week_day` :

first_week_day values

Jour	Valeur
Lundi	0

Jour	Valeur
Mardi	1
Mercredi	2
Jeudi	3
Vendredi	4
Samedi	5
Dimanche	6

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
weekname('01/12/2013')	Renvoie 2013/02.
weekname('01/12/2013', -1)	Returns 2013/01.
weekname('01/12/2013', 0, 1)	Renvoie 2013/02.

Exemple 1 – date sans aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2021 et les deux premières semaines de 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `DateFormat` définie au format `MM/DD/YYYY`.
- Variable système `BrokenWeeks` définie sur `1`.
- Variable système `FirstWeekDay` définie sur `6`.
- Chargement précédent contenant les éléments suivants :
    - La fonction `weekday()`, qui est définie comme le champ, '`week_number`', qui renvoie l'année et le numéro de semaine de réalisation des transactions.
    - La fonction `weekname()`, qui est définie comme le champ appelé '`week_day`', pour afficher la valeur weekday de chaque date de transaction.

Script de chargement

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     WeekDay(date) as week_day,
     Weekname(date) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- week_day
- week_number

Tableau de résultats

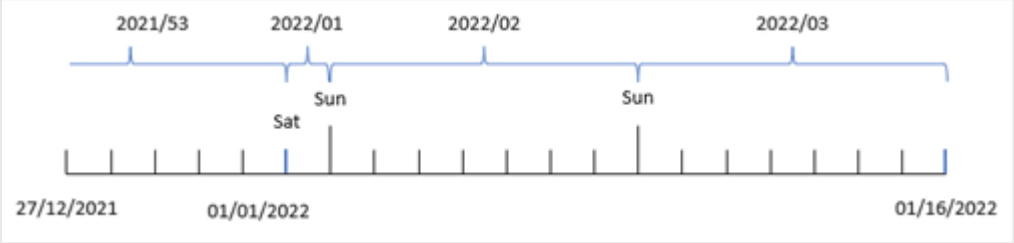
id	date	week_day	week_number
8183	12/27/2021	Mon	2021/53
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Le champ '`week_number`' est créé dans l'instruction LOAD précédente via la fonction `weekname()` et en transmettant le champ date comme argument de la fonction.

La fonction `weekname()` identifie initialement la semaine dans laquelle tombe la valeur date et renvoie le nombre de semaines et l'année de la transaction.

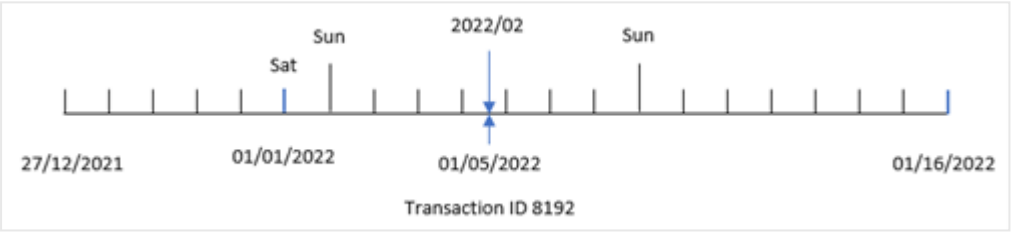
La variable système `FirstWeekDay` définit le dimanche comme le premier jour de la semaine. La variable système `BrokenWeeks` définit l'application de sorte qu'elle utilise des semaines interrompues, ce qui signifie que la semaine 1 commencera le 1er janvier.

Diagramme de la fonction `weekname()` avec les variables par défaut.



La semaine 1 commence le 1er janvier, qui est un samedi, et, par conséquent, les transactions effectuées à cette date renvoient la valeur `2022/01` (l'année et le numéro de semaine).

Diagramme de la fonction `weekname()` identifiant le numéro de semaine de la transaction 8192.



Étant donné que l'application utilise des semaines interrompues et que le premier jour de la semaine est le dimanche, les transactions effectuées du 2 au 8 janvier renvoient la valeur `2022/02` (semaine numéro 2 en 2022). Un exemple de ceci serait la transaction 8192, qui a eu lieu le 5 janvier et qui renvoie la valeur `2022/02` pour le champ '`week_number`'.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la tâche consiste à créer un champ, '`previous_week_number`', qui renvoie l'année et le numéro de semaine précédant la réalisation des transactions.

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement suivant à un nouvel onglet.

Script de chargement

```
SET BrokenWeeks=1;
SET FirstWeekDay=6;

Transactions:
   Load
```

```
     *,
     weekname(date,-1) as previous_week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- week_day
- week_number

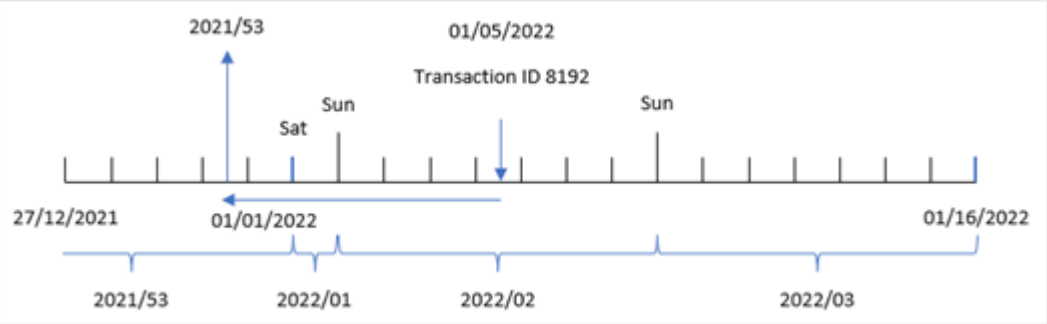
Tableau de résultats

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/52
8185	12/29/2021	Wed	2021/52
8186	12/30/2021	Thu	2021/52
8187	12/31/2021	Fri	2021/52
8188	01/01/2022	Sat	2021/52

id	date	week_day	week_number
8189	01/02/2022	Sun	2021/53
8190	01/03/2022	Mon	2021/53
8191	01/04/2022	Tue	2021/53
8192	01/05/2022	Wed	2021/53
8193	01/06/2022	Thu	2021/53
8194	01/07/2022	Fri	2021/53
8195	01/08/2022	Sat	2022/01
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02
8198	01/11/2022	Tue	2022/02
8199	01/12/2022	Wed	2022/02
8200	01/13/2022	Thu	2022/02
8201	01/14/2022	Fri	2022/02

Étant donné qu'une valeur `period_no` de `-1` est utilisée comme argument de décalage dans la fonction `weekname()`, la fonction commence par identifier la semaine au cours de laquelle les transactions ont lieu. Elle regarde ensuite une semaine avant et identifie la première milliseconde de cette semaine.

Diagramme de la fonction `weekname()` avec un décalage `period_no` de -1.



La transaction 8192 a eu lieu le 5 janvier 2022. La fonction `weekname()` recherche une semaine avant, le 30 décembre 2021, et renvoie le numéro de semaine et l'année de cette date : `2021/53`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la politique de l'entreprise stipule que la semaine de travail commence le mardi.

Ouvrez l'Éditeur de chargement de données et ajoutez le script de chargement suivant à un nouvel onglet.

Script de chargement

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';

Transactions:
   Load
     *,
     weekday(date) as week_day,
     weekname(date,0,1) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

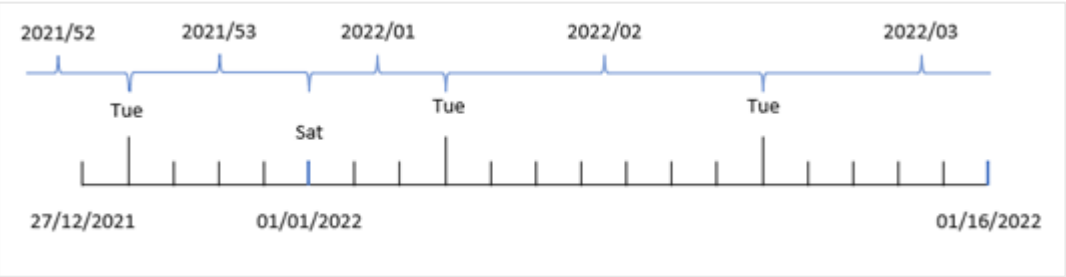
Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- week_day
- week_number

Tableau de résultats

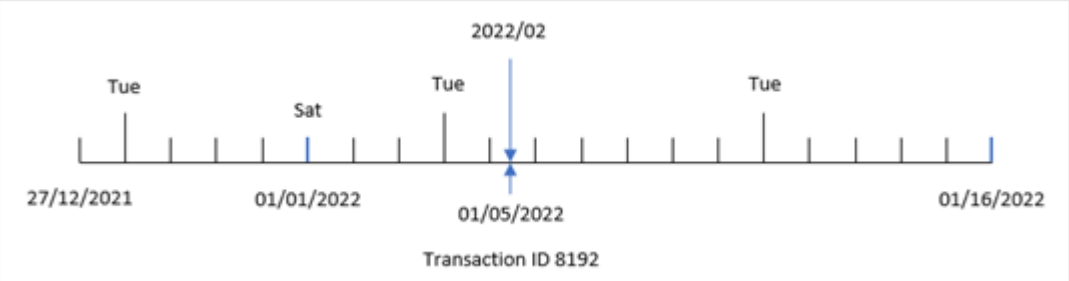
id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/01
8190	01/03/2022	Mon	2022/01
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Diagramme de la fonction `weekname()` avec mardi comme premier jour de la semaine.



Étant donné que l'argument `first_week_date` égal à `1` est utilisé dans la fonction `weekname()`, le mardi est utilisé comme le premier jour de la semaine. La fonction détermine donc que la semaine 53 2021 commence le mardi 28 décembre ; et, du fait que l'application utilise des semaines interrompues, la semaine 1 commence le 1er janvier 2022 et se termine la dernière milliseconde du lundi 3 janvier 2022.

Diagramme montrant le numéro de semaine de la transaction 8192 avec le mardi comme premier jour de la semaine.



La transaction 8192 a eu lieu le 5 janvier 2022. Par conséquent, avec l'utilisation d'un paramètre `first_week_day` défini sur Tuesday (mardi), la fonction `weekname()` renvoie la valeur `2022/02` pour le champ '`week_number`'.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui renvoie l'année de la semaine de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
SET BrokenWeeks=1;
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
```

```
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `=week_day (date)`

Pour calculer le début de la semaine d'une transaction, créez la mesure suivante :

```
=weekname(date)
```

Tableau de résultats

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Mon	2021/53
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Le champ 'week_number' est créé sous forme de mesure dans l'objet graphique via la fonction weekname() et en transmettant le champ date comme argument de la fonction.

La fonction weekname() identifie initialement la semaine dans laquelle tombe la valeur date et renvoie le nombre de semaines et l'année de la transaction.

La variable système FirstWeekDay définit le dimanche comme le premier jour de la semaine. La variable système BrokenWeeks définit l'application de sorte qu'elle utilise des semaines interrompues, ce qui signifie que la semaine 1 commence le 1er janvier.

Diagramme montrant le numéro de semaine avec le dimanche comme le premier jour de la semaine.

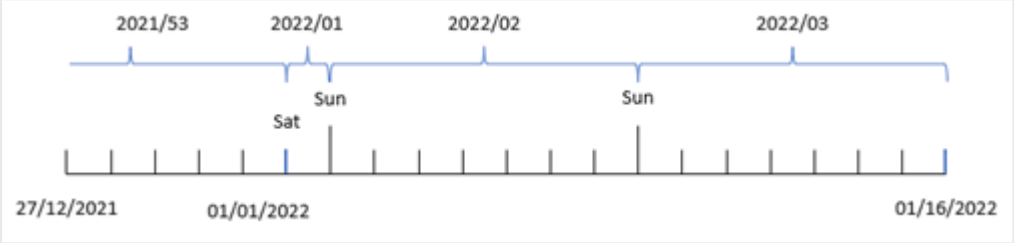
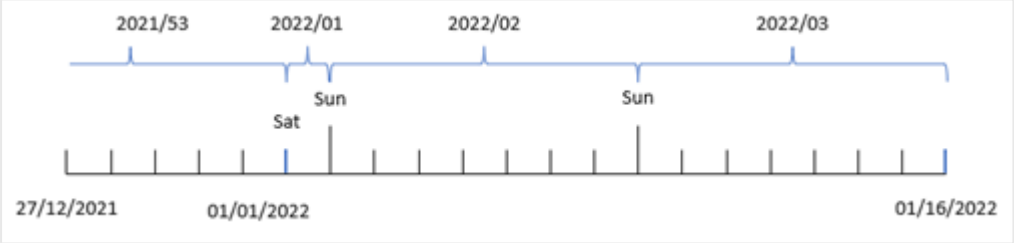


Diagramme montrant que la transaction 8192 a eu lieu au cours de la semaine numéro deux.



Étant donné que l'application utilise des semaines interrompues et que le premier jour de la semaine est le dimanche, les transactions effectuées du 2 au 8 janvier renvoient la valeur 2022/02, à savoir, la semaine numéro 2 de 2022. Notez que la transaction 8192 a eu lieu le 5 janvier et qu'elle renvoie la valeur 2022/02 pour le champ 'week_number'.

Exemple 5 – scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2019 et les deux premières semaines de 2020, chargé dans une table appelée 'Transactions'.
- Variable système BrokenWeeks définie sur 0.

- Variable système `ReferenceDay` définie sur `2`.
- Variable système `DateFormat` définie au format `MM/DD/YYYY`.

Script de chargement

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez un tableau.

Créez une dimension calculée à l'aide de l'expression suivante :

```
=weekname(date)
```

Pour calculer les ventes totales, créez la mesure d'agrégation suivante :

```
=sum(amount)
```

Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

Pour démontrer les résultats de l'utilisation de la fonction weekname() dans ce scénario, ajoutez le champ suivant comme dimension :

```
date
```

Tableau de résultats avec le champ date

weekname(date)	date	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23
2020/01	01/02/2020	$17.17
2020/01	01/03/2020	$88.27
2020/01	01/04/2020	$57.42
2020/02	01/05/2020	$53.80
2020/02	01/06/2020	$82.06
2020/02	01/07/2020	$40.56
2020/02	01/08/2020	$53.67
2020/02	01/09/2020	$26.63
2020/02	01/10/2020	$72.48
2020/02	01/11/2020	$18.37
2020/03	01/12/2020	$45.26
2020/03	01/13/2020	$58.23
2020/03	01/14/2020	$18.52

Étant donné que l'application utilise des semaines ininterrompues et que la semaine 1 nécessite un minimum de deux jours en janvier en raison de la variable système `ReferenceDay`, la semaine 1 2020 inclut les transactions du 29 décembre 2019.

weekstart

Cette fonction renvoie une valeur correspondant à un horodatage de la première milliseconde du premier jour de la semaine calendaire contenant l'argument **date**. Le format de sortie par défaut correspond au format de date **DateFormat** défini dans le script.

Syntaxe :

```
WeekStart(date [, period_no[, first_week_day]])
```

Type de données renvoyé : double

Diagramme de la fonction `weekstart()`

La fonction `weekstart()` détermine la semaine au cours de laquelle tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la première milliseconde de cette semaine-là. Le premier jour de la semaine est déterminé par la variable d'environnement `FirstWeekDay`. Cependant, cela peut être remplacé par l'argument `first_week_day` dans la fonction `weekstart()`.

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**shift** est un entier, où la valeur 0 indique la semaine comprenant l'argument **date**. Les valeurs négatives spécifiées pour shift indiquent les semaines passées tandis que les valeurs positives désignent les semaines à venir.
first_week_day	Spécifie le jour où débute la semaine. S'il est omis, c'est la valeur de la variable **FirstWeekDay** qui est utilisée. Les valeurs possibles de **first_week_day** sont 0 pour lundi, 1 pour mardi, 2 pour mercredi, 3 pour jeudi, 4 pour vendredi, 5 pour samedi et 6 pour dimanche. Pour plus d'informations sur la variable système, voir *FirstWeekDay (page 221)*.

Cas d'utilisation

La fonction `weekstart()` est couramment utilisée dans le cadre d'une expression lorsque l'utilisateur souhaite que le calcul utilise la fraction de la semaine qui s'est écoulée jusqu'à présent. Par exemple, cela peut permettre à un utilisateur de calculer le total des salaires touchés par les employés au cours de la semaine jusqu'à ce jour.

Exemples de fonction

Exemple	Résultat
`weekstart('01/12/2013')`	Renvoie 01/07/2013.
`weekstart('01/12/2013', -1 )`	Renvoie 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Renvoie 01/08/2013.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `start_of_week`, qui renvoie un horodatage du début de la semaine au cours de laquelle les transactions ont eu lieu.

Script de chargement

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekstart(date) as start_of_week,
```

```
        timestamp(weekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- date
- start_of_week
- start_of_week_timestamp

Tableau de résultats

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/17/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

Le champ `start_of_week` est créé dans l'instruction LOAD précédente via la fonction `weekstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `weekstart()` identifie initialement la semaine de la valeur date, renvoyant un horodatage pour la première milliseconde de cette semaine-là.

Diagramme de la fonction `weekstart()`, exemple sans argument supplémentaire

La transaction 8191 a eu lieu le 5 février. La variable système `FirstWeekDay` définit le premier jour de la semaine comme étant un dimanche. La fonction `weekstart()` identifie que le premier dimanche avant le 5 février – et par conséquent le début de la semaine – était le 30 janvier. Par conséquent, la valeur `start_of_week` pour cette transaction renvoie la première milliseconde de ce jour-là, le 30 janvier à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `previous_week_start`, qui renvoie l'horodatage du début du trimestre avant la transaction.

Script de chargement

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

Transactions:
    Load
        *,
        weekstart(date,-1) as previous_week_start,
        timestamp(weekstart(date,-1)) as previous_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

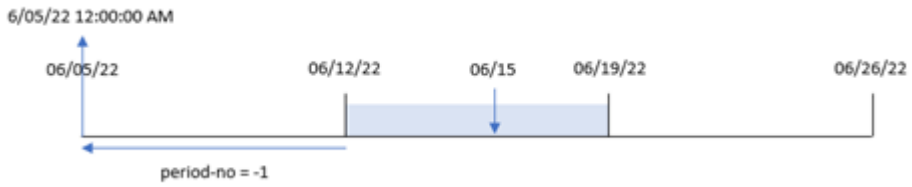
- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

Tableau de résultats

date	previous_week_start	previous_week_start_timestamp
1/7/2022	12/26/2021	12/26/2021 12:00:00 AM
1/19/2022	01/09/2022	1/9/2022 12:00:00 AM
2/5/2022	01/23/2022	1/23/2022 12:00:00 AM
2/28/2022	02/20/2022	2/20/2022 12:00:00 AM
3/16/2022	03/06/2022	3/6/2022 12:00:00 AM
4/1/2022	03/20/2022	3/20/2022 12:00:00 AM
5/7/2022	04/24/2022	4/24/2022 12:00:00 AM
5/16/2022	05/08/2022	5/8/2022 12:00:00 AM
6/15/2022	06/05/2022	6/5/2022 12:00:00 AM
6/26/2022	06/19/2022	6/19/2022 12:00:00 AM
7/9/2022	06/26/2022	6/26/2022 12:00:00 AM
7/22/2022	07/10/2022	7/10/2022 12:00:00 AM
7/23/2022	07/10/2022	7/10/2022 12:00:00 AM
7/27/2022	07/17/2022	7/17/2022 12:00:00 AM
8/2/2022	07/24/2022	7/17/2022 12:00:00 AM
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/7/2022 12:00:00 AM
9/26/2022	09/18/2022	9/18/2022 12:00:00 AM
10/14/2022	10/02/2022	10/2/2022 12:00:00 AM
10/29/2022	10/16/2022	10/16/2022 12:00:00 AM

Dans cet exemple, étant donné que la valeur `period_no` de -1 a été utilisée comme argument de décalage dans la fonction `weekstart()`, la fonction commence par identifier la semaine au cours de laquelle les transactions ont lieu. Elle regarde ensuite une semaine avant et identifie la première milliseconde de cette semaine.

Diagramme de la fonction `weekstart()`, exemple period_no



La transaction 8196 a eu lieu le 15 juin. La fonction `weekstart()` identifie que la semaine commence le 12 juin. Par conséquent, la semaine précédente a commencé le 5 juin à 12:00:00 AM ; il s'agit de la valeur renvoyée pour le champ `previous_week_start`.

Exemple 3 – first_week_day

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple. Cependant, dans cet exemple, nous devons définir le mardi comme le premier jour de la semaine de travail.

Script de chargement

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

Transactions:
    Load
        *,
        weekstart(date,0,1) as start_of_week,
        timestamp(weekstart(date,0,1)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
```

```
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tableau de résultats

date	start_of_week	start_of_week_timestamp
1/7/2022	01/04/2022	1/4/2022 12:00:00 AM
1/19/2022	01/18/2022	1/18/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/22/2022	2/22/2022 12:00:00 AM
3/16/2022	03/15/2022	3/15/2022 12:00:00 AM
4/1/2022	03/29/2022	3/29/2022 12:00:00 AM
5/7/2022	05/03/2022	5/3/2022 12:00:00 AM
5/16/2022	05/10/2022	5/10/2022 12:00:00 AM
6/15/2022	06/14/2022	6/14/2022 12:00:00 AM
6/26/2022	06/21/2022	6/21/2022 12:00:00 AM
7/9/2022	07/05/2022	7/5/2022 12:00:00 AM
7/22/2022	07/19/2022	7/19/2022 12:00:00 AM
7/23/2022	07/19/2022	7/19/2022 12:00:00 AM
7/27/2022	07/26/2022	7/26/2022 12:00:00 AM
8/2/2022	08/02/2022	8/2/2022 12:00:00 AM
8/8/2022	08/02/2022	8/2/2022 12:00:00 AM
8/19/2022	08/16/2022	8/16/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
9/26/2022	09/20/2022	9/20/2022 12:00:00 AM
10/14/2022	10/11/2022	10/11/2022 12:00:00 AM
10/29/2022	10/25/2022	10/25/2022 12:00:00 AM

Dans ce cas, étant donné que l'argument `first_week_date` égal à 1 est utilisé dans la fonction `weekstart ()`, il définit le premier jour de la semaine comme étant le mardi.

Diagramme de la fonction `weekstart()`, exemple first_week_day

La transaction 8191 a eu lieu le 5 février. La fonction `weekstart()` identifie que le premier mardi avant cette date – et par conséquent le début de la semaine et la valeur renvoyée – était le 1er février à 12:00:00 AM.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui renvoie un horodatage correspondant au début de la semaine de réalisation des transactions est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
```

```
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Pour calculer le début de la semaine d'une transaction, ajoutez les mesures suivantes :

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Tableau de résultats

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/17/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

La mesure `start_of_week` est créée dans l'objet graphique via la fonction `weekstart()` et en transmettant le champ `date` comme argument de la fonction.

La fonction `weekstart()` identifie initialement la semaine de la valeur date, renvoyant un horodatage pour la première milliseconde de cette semaine-là.

Diagramme de la fonction `weekstart()`, exemple objet graphique

La transaction 8191 a eu lieu le 5 février. La variable système `FirstWeekDay` définit le premier jour de la semaine comme étant un dimanche. La fonction `weekstart()` identifie que le premier dimanche avant le 5 février – et par conséquent le début de la semaine – était le 30 janvier. Par conséquent, la valeur `start_of_week` de cette transaction renvoie la première milliseconde de ce jour-là, à savoir, le 30 janvier à 12:00:00 AM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée `Payroll`.
- Données constituées des ID des employés, des noms des employés et du salaire quotidien touché par chaque employé.

Les employés commencent à travailler le lundi et travaillent six jours par semaine. La variable système `FirstWeekDay` ne doit pas être modifiée.

L'utilisateur final souhaite un objet graphique qui affiche, par ID d'employé et nom d'employé, les salaires touchés au cours de la semaine jusqu'à la date du jour.

Script de chargement

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Résultats

Procédez comme suit :

1. Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :
   - `employee_id`
   - `employee_name`
2. Ensuite, créez une mesure pour calculer les salaires touchés au cours de la semaine jusqu'à la date du jour :
   `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Définissez le **Formatage des nombres** des mesures sur **Devise**.

Tableau de résultats

employee_id	employee_name	=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)
182	Mark	$600.00
183	Deryck	$500.00
184	Dexter	$500.00
185	Sydney	$1080.00
186	Agatha	$512.00

La fonction `weekstart()`, en utilisant la date d'aujourd'hui comme premier argument et 0 comme troisième argument, définit le lundi comme le premier jour de la semaine et renvoie la date de début de la semaine en cours. En soustrayant ce résultat de la date actuelle, l'expression renvoie alors le nombre de jours qui se sont écoulés jusqu'à présent cette semaine.

La condition évalue ensuite si cette semaine comptait plus de six jours. Si c'est le cas, la valeur `day_rate` de l'employé est multipliée par 6 jours. Sinon, la valeur `day_rate` est multipliée par le nombre de jours qui se sont déroulés jusqu'à ce jour cette semaine.

weekyear

Cette fonction renvoie l'année à laquelle le numéro de semaine appartient selon la norme ISO 8601. Le numéro de la semaine est compris entre 1 et environ 52.

Syntaxe :

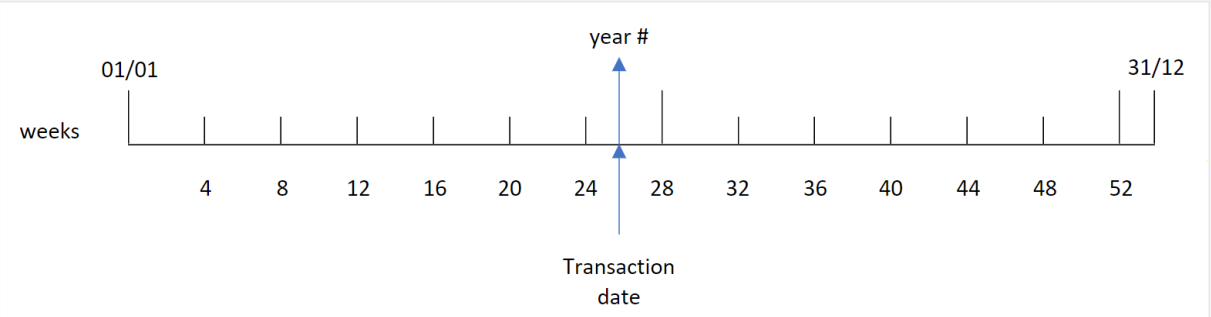
```
weekyear(expression)
```

Type de données renvoyé : entier

La fonction `weekyear()` détermine la semaine d'une année au cours de laquelle tombe une date. Elle renvoie ensuite l'année correspondant à ce numéro de semaine.

Par défaut, les applications Qlik utilisent des semaines interrompues (définies par la variable système `BrokenWeeks`) et le numéro de semaine 1 commence le 1er janvier et l'année se termine à la fin de la semaine 52. Par conséquent, la fonction `weekyear()` renverra toujours la même valeur que la fonction `week()` lorsque l'application utilise des semaines interrompues.

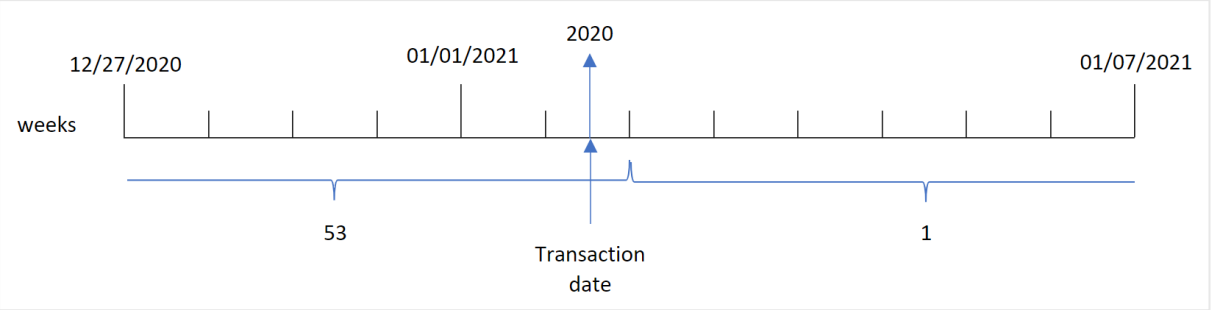
Diagramme de la plage de la fonction `weekyear()`



Cependant, si la variable système `BrokenWeeks` est définie de sorte à utiliser des semaines ininterrompues, la semaine 1 ne doit contenir qu'un certain nombre de jours en janvier, suivant la valeur spécifiée dans la variable système `ReferenceDay`.

Par exemple, si une valeur `ReferenceDay` égale à `4` est utilisée, la semaine 1 doit inclure au moins quatre jours en janvier. Il est possible que la semaine 1 inclue des dates en décembre de l'année précédente ou que le dernier numéro de semaine d'une année inclue des dates en janvier de l'année suivante. Dans des situations telles que celle-ci, la fonction `weekyear()` renverra une valeur différente de celle de la fonction `year()`.

Diagramme de la plage de la fonction `weekyear()` lors de l'utilisation de semaines ininterrompues



Cas d'utilisation

La fonction `weekyear()` est utile lorsque vous souhaitez comparer des agrégations par année. Par exemple, si vous souhaitez voir les ventes totales de produits par année. La fonction `weekyear()` est préférée à la fonction `year()` lorsque l'utilisateur souhaite maintenir la cohérence avec la variable système `BrokenWeeks` dans l'application.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`weekyear('12/30/1996')`	Renvoie 1997, car la semaine 1 de 1997 commence le 12/30/1996.
`weekyear('01/02/1997')`	Renvoie 1997.
`weekyear('12/28/1997')`	Renvoie 1997.
`weekyear('12/30/1997')`	Renvoie 1998, car la semaine 1 de 1998 commence le 12/29/1997.
`weekyear('01/02/1999')`	Renvoie 1998, car la semaine 53 de 1998 se termine le 01/03/1999.

Rubriques connexes

Rubrique	Interaction
week (page 1056)	Renvoie un entier représentant le numéro de la semaine selon la norme ISO 8601.
year (page 1130)	Renvoie un entier représentant l'année au cours de laquelle l'expression est interprétée comme une date selon l'interprétation standard des nombres.

Exemple 1 - semaines interrompues

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2020 et la première semaine de 2021, chargé dans une table appelée '`Transactions`'.
- Variable `BrokenWeeks` définie sur 1.
- Chargement précédent contenant les éléments suivants :
  - Fonction `weekyear()`, définie comme le champ '`week_year`', qui renvoie l'année de réalisation des transactions.
  - Fonction `week()`, définie comme le champ '`week`', qui indique le numéro de semaine de chaque date de transaction.

Script de chargement

```
SET BrokenWeeks=1;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
```

```
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `week`
- `week_year`

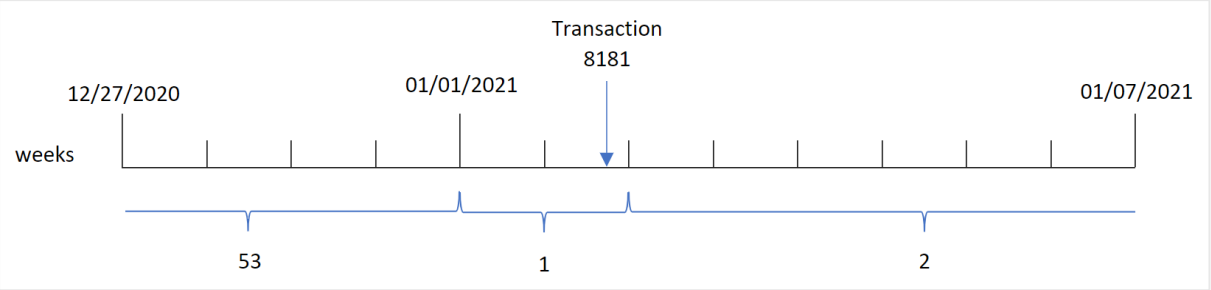
Tableau de résultats

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Le champ '`week_year`' est créé dans l'instruction LOAD précédente via la fonction `weekyear()` et en transmettant le champ date comme argument de la fonction.

La variable système `BrokenWeeks` est définie sur `1`, ce qui signifie que l'application utilise des semaines interrompues. La semaine 1 commence le 1er janvier.

Diagramme de la plage de la fonction `weekyear()` lors de l'utilisation de semaines interrompues



La transaction 8181 a lieu le 2 janvier, qui fait partie de la semaine 1. Par conséquent, elle renvoie une valeur 2021 pour le champ 'week_year'.

Exemple 2 - semaines ininterrompues

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2020 et la première semaine de 2021, chargé dans une table appelée 'Transactions'.
- Variable `BrokenWeeks` définie sur `0`.
- Chargement précédent contenant les éléments suivants :
  - Fonction `weekyear()`, définie comme le champ 'week_year', qui renvoie l'année de réalisation des transactions.
  - Fonction `week()`, définie comme le champ 'week', qui indique le numéro de semaine de chaque date de transaction.

Cependant, dans cet exemple, la stratégie de l'entreprise exige d'utiliser des semaines ininterrompues.

Script de chargement

```
SET BrokenWeeks=0;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
```

```
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `week`
- `week_year`

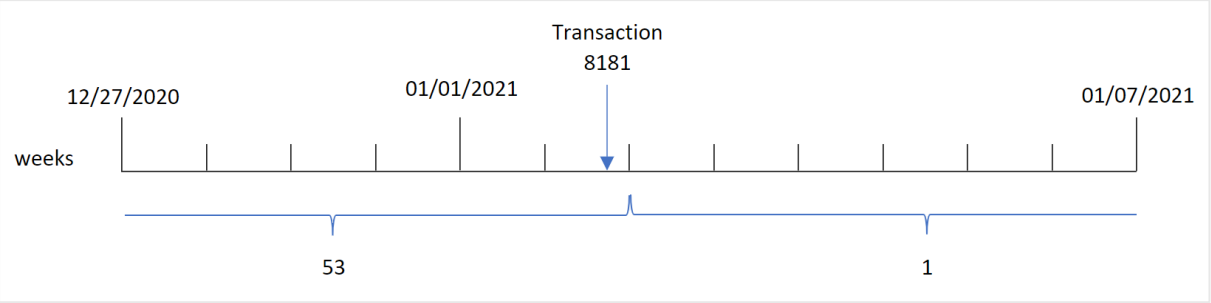
Tableau de résultats

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	53	2020
8181	01/02/2021	53	2020
8182	01/03/2021	1	2021
8183	01/04/2021	1	2021
8184	01/05/2021	1	2021
8185	01/06/2021	1	2021
8186	01/07/2021	1	2021

La variable système `BrokenWeeks` est définie sur `0`, ce qui signifie que l'application utilise des semaines ininterrompues. Par conséquent, la semaine 1 ne commence pas obligatoirement le 1er janvier.

La semaine 53 2020 se poursuit jusqu'à la fin du 2 janvier 2021, la semaine 1 2020 commençant quant à elle le dimanche 3 janvier 2021.

Diagramme de la plage de la fonction `weekyear()` lors de l'utilisation de semaines ininterrompues



La transaction 8181 a lieu le 2 janvier, qui fait partie de la semaine 1. Par conséquent, elle renvoie une valeur 2021 pour le champ '`week_year`'.

Exemple 3 - exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui renvoie le numéro de semaine de l'année de réalisation des transactions est créé sous forme de mesure dans un graphique de l'application.

Script de chargement

```
SET BrokenWeeks=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date

Pour calculer la semaine d'une transaction, créez la mesure suivante :

- =week(date)

Pour calculer l'année d'une transaction en fonction du numéro de semaine, créez la mesure suivante :

- =weekyear(date)

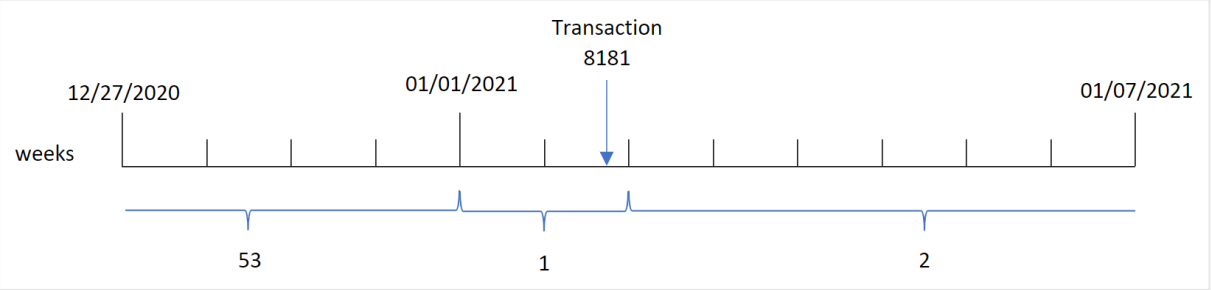
Tableau de résultats

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Le champ 'week_year' est créé dans l'instruction LOAD précédente via la fonction weekyear() et en transmettant le champ date comme argument de la fonction.

La variable système BrokenWeeks est définie sur 1, ce qui signifie que l'application utilise des semaines interrompues. La semaine 1 commence le 1er janvier.

Diagramme de la plage de la fonction weekyear() lors de l'utilisation de semaines interrompues



La transaction 8181 a lieu le 2 janvier, qui fait partie de la semaine 1. Par conséquent, elle renvoie une valeur 2021 pour le champ 'week_year'.

Exemple 4 - scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions pour la dernière semaine de 2020 et la première semaine de 2021, chargé dans une table appelée '`Transactions`'.
- Variable `BrokenWeeks` définie sur `0`. Cela signifie que l'application utilisera des semaines ininterrompues.
- Variable `ReferenceDay` définie sur `2`. Cela signifie que l'année commencera le 2 janvier et contiendra au moins deux jours en janvier.
- Variable `FirstWeekDay` définie sur `1`. Cela signifie que le premier jour de la semaine sera un mardi.

La stratégie de l'entreprise consiste à utiliser des semaines interrompues. L'utilisateur final souhaite un graphique présentant les ventes totales par année. L'application utilise des semaines ininterrompues avec la semaine 1 contenant au moins deux jours en janvier.

Script de chargement

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET FirstWeekDay=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez un tableau.

Pour calculer l'année d'une transaction en fonction du numéro de semaine, créez la mesure suivante :

- `=weekyear(date)`

Pour calculer les ventes totales, créez la mesure suivante :

- `sum(amount)`

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Cette fonction renvoie un entier représentant l'année au cours de laquelle l'**expression** est interprétée comme une date selon l'interprétation standard des nombres.

Syntaxe :

```
year(expression)
```

Type de données renvoyé : entier

La fonction `year()` est disponible comme fonction de script et de graphique. La fonction renvoie l'année d'une date donnée. Elle est couramment utilisée pour créer un champ year comme dimension dans un Master Calendar.

Cas d'utilisation

La fonction `year()` est utile lorsque vous souhaitez comparer des agrégations par année. Par exemple, la fonction peut être utilisée si vous souhaitez voir les ventes totales de produits par année.

Il est possible de créer ces dimensions dans le script de chargement via la fonction permettant de créer un champ dans une table Master Calendar. Sinon, elle peut également être directement utilisée dans un graphique comme dimension calculée.

Exemples de fonction

Exemple	Résultat
`year( '2012-10-12' )`	renvoie 2012.
`year( '35648' )`	renvoie 1997, car 35648 = 1997-08-06.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux

et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – Ensemble de données DateFormat (script)

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, chargé dans une table appelée `Master Calendar`.
- La variable système `DateFormat` par défaut (MM/DD/YYYY) est utilisée.
- Chargement précédent, utilisé pour créer un champ supplémentaire, `year`, via la fonction `year()`.

Script de chargement

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

Master_Calendar:

    Load
        date,
        year(date) as year
        ;
Load
date
Inline
[
date
12/28/2020
12/29/2020
12/30/2020
12/31/2020
01/01/2021
01/02/2021
01/03/2021
01/04/2021
01/05/2021
01/06/2021
```

```
01/07/2021
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `year`

Tableau de résultats

date	année
12/28/2020	2020
12/29/2020	2020
12/30/2020	2020
12/31/2020	2020
01/01/2021	2021
01/02/2021	2021
01/03/2021	2021
01/04/2021	2021
01/05/2021	2021
01/06/2021	2021
01/07/2021	2021

Exemple 2 – dates ANSI

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, chargé dans une table appelée `Master Calendar`.
- La variable système `DateFormat` par défaut (MM/DD/YYYY) est utilisée. Cependant, les dates incluses dans l'ensemble de données sont au format de date standard ANSI.
- Chargement précédent, utilisé pour créer un champ supplémentaire, appelé `year`, via la fonction `year()`.

Script de chargement

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

Master_Calendar:
    Load
        date,
        year(date) as year
        ;
Load
date
Inline
[
date
2020-12-28
2020-12-29
2020-12-30
2020-12-31
2021-01-01
2021-01-02
2021-01-03
2021-01-04
2021-01-05
2021-01-06
2021-01-07
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `year`

Tableau de résultats

date	année
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021

date	année
2021-01-06	2021
2021-01-07	2021

Exemple 3 – dates non formatées

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données de dates, au format numérique, chargé dans une table appelée `Master Calendar`.
- La variable système `DateFormat` par défaut (MM/DD/YYYY) est utilisée.
- Chargement précédent, utilisé pour créer un champ supplémentaire, `year`, via la fonction `year()`.

La date non formatée originale est chargée, appelée `unformatted_date`, et, pour clarifier les choses, un autre champ supplémentaire, appelé `long_date`, est utilisé pour convertir la date numérique en champ de date formatée via la fonction `date()`.

Script de chargement

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

Master_Calendar:
    Load
        unformatted_date,
        date(unformatted_date) as long_date,
        year(unformatted_date) as year
        ;
Load
unformatted_date
Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
```

```
45068
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `unformatted_date`
- `long_date`
- `year`

Tableau des résultats

unformatted_date	long_date	année
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023
44958	02/01/2023	2023
44988	03/03/2023	2023
45008	03/23/2023	2023
45018	04/02/2023	2023
45038	04/22/2023	2023
45048	05/02/2023	2023
45068	05/22/2023	2023
45078	06/01/2023	2023

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Dans cet exemple, un ensemble de données de commandes passées est chargé dans une table nommée Sales. La table contient trois champs :

- `id`
- `sales_date`
- `amount`

Les garanties sur les ventes de produits au cours des deux dernières années à compter de la date de vente. La tâche consiste à créer une mesure dans un graphique pour déterminer l'année d'expiration de chaque garantie.

Script de chargement

```
Sales:
Load
id,
sales_date,
amount
Inline
[
id,sales_date,amount
1,12/28/2020,231.24,
2,12/29/2020,567.28,
3,12/30/2020,364.28,
4,12/31/2020,575.76,
5,01/01/2021,638.68,
6,01/02/2021,785.38,
7,01/03/2021,967.46,
8,01/04/2021,287.67
9,01/05/2021,764.45,
10,01/06/2021,875.43,
11,01/07/2021,957.35
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `sales_date`.

Créez la mesure suivante :

```
=year(sales_date+365*2)
```

Tableau de résultats

sales_date	=year(sales_date+365*2)
12/28/2020	2022
12/29/2020	2022
12/30/2020	2022
12/31/2020	2022
01/01/2021	2023
01/02/2021	2023
01/03/2021	2023
01/04/2021	2023

sales_date	=year(sales_date+365*2)
01/05/2021	2023
01/06/2021	2023
01/07/2021	2023

Les résultats de cette mesure sont affichés dans le tableau ci-dessus. Pour ajouter deux années à une date, multipliez 365 par 2 et ajoutez le résultat à la date de vente. Par conséquent, la date d'expiration des ventes qui ont eu lieu en 2020 est 2022.

yearend

Cette fonction renvoie une valeur correspondant à un horodatage de la dernière milliseconde du dernier jour de l'année contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

En d'autres termes, la fonction `yearend()` détermine l'année dans laquelle tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la dernière milliseconde de l'année en question. Le premier mois de l'année est, par défaut, janvier. Cependant, vous pouvez modifier le mois défini comme le premier en utilisant l'argument `first_month_of_year` dans la fonction `yearend()`.

> *La fonction `yearend()` ne prend pas en compte la variable système `FirstMonthOfYear`. L'année commence le 1er janvier, sauf si l'argument `first_month_of_year` est utilisé pour modifier cela.*

Diagramme de la fonction `yearend()`.

Cas d'utilisation

La fonction `yearend()` est utilisée dans le cadre d'une expression lorsque vous souhaitez que le calcul utilise la fraction de l'année qui n'a pas encore eu lieu. Par exemple, si vous souhaitez calculer le total des intérêts non encore encourus au cours de l'année.

Type de données renvoyé : double

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique l'année comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les années passés tandis que les valeurs positives désignent les années à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`yearend('10/19/2001')`	Renvoie 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Renvoie 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Renvoie 03/31/2002 23:59:59.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Champ de date fourni dans la variable système `DateFormat` au format `(MM/DD/YYYY)`.
- Instruction LOAD précédente contenant les éléments suivants :
  - La fonction `yearend()` définie comme le champ `year_end`.
  - La fonction `Timestamp()` définie comme le champ `year_end_timestamp`.

Script de chargement

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

Transactions:
    Load
       *,
       yearend(date) as year_end,
       timestamp(yearend(date)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
```

```
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- year_end
- year_end_timestamp

Tableau de résultats

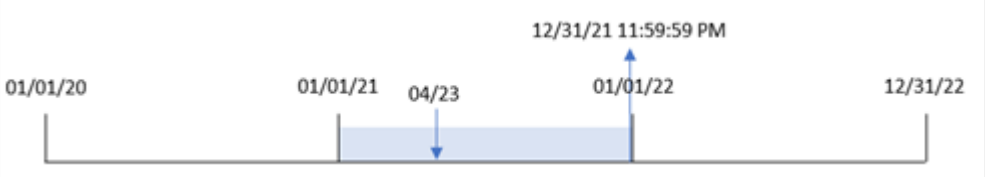
id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM

id	date	year_end	year_end_timestamp
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

Le champ '`year_end`' est créé dans l'instruction LOAD précédente via la fonction `yearend()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearend()` identifie initialement l'année dans laquelle tombe la valeur date et renvoie un horodatage pour la dernière milliseconde de cette année.

Diagramme de la fonction `yearend()` avec la transaction 8199 sélectionnée.



La transaction 8199 a eu lieu le 23 avril 2021. La fonction `yearend()` renvoie la dernière milliseconde de cette année, soit le 31 décembre à 11:59:59 PM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la tâche consiste à créer un champ, '`previous_year_end`', qui renvoie l'horodatage de la date de fin de l'année précédant l'année de la transaction.

Script de chargement

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

Transactions:
```

```
    Load
       *,
       yearend(date,-1) as previous_year_end,
       timestamp(yearend(date,-1)) as previous_year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- previous_year_end
- previous_ year_end_timestamp

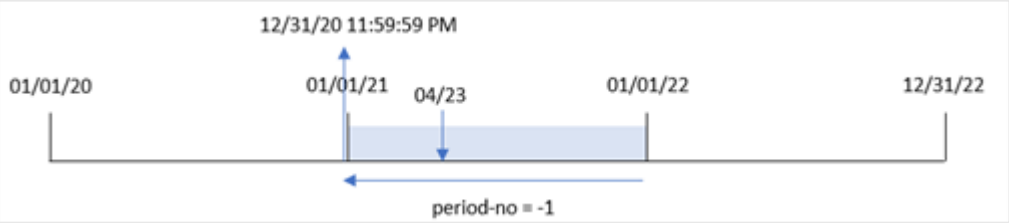
Tableau de résultats

id	date	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM

id	date	previous_year_end	previous_year_end_timestamp
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

Étant donné qu'une valeur `period_no` de `-1` a été utilisée comme argument de décalage dans la fonction `yearend()`, la fonction commence par identifier l'année au cours de laquelle les transactions ont lieu. Elle regarde ensuite une année avant et identifie la dernière milliseconde de cette année.

Diagramme de la fonction `yearend()` avec une valeur period_no de -1.



La transaction 8199 a lieu le 23 avril 2021. La fonction `yearend()` renvoie la dernière milliseconde de l'année précédente, soit le 31 décembre 2020 à 11:59:59 PM, pour le champ '`previous_year_end`'.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la politique de l'entreprise stipule que l'année commence le 1er avril.

Script de chargement

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

Transactions:
    Load
       *,
       yearend(date,0,4) as year_end,
       timestamp(yearend(date,0,4)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

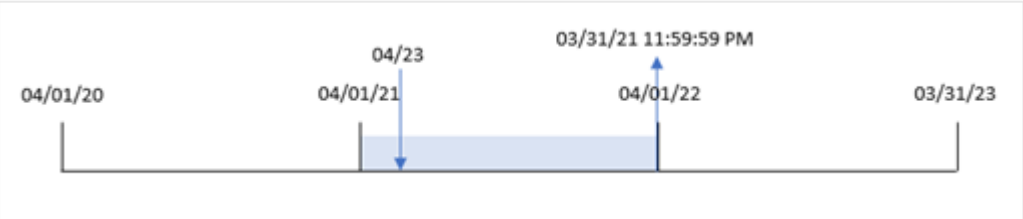
- id
- date
- year_end
- year_end_timestamp

Tableau de résultats

id	date	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

Étant donné que l'argument `first_month_of_year` égal à `4` est utilisé dans la fonction `yearend()`, le premier jour de l'année est défini sur le 1er avril et le dernier jour de l'année sur le 31 mars.

Diagramme de la fonction `yearend()` avec avril comme premier mois de l'année.



La transaction 8199 a lieu le 23 avril 2021. Étant donné que la fonction `yearend()` définit le début de l'année sur le 1er avril, elle renvoie le 31 mars 2022 comme valeur '`year_end`' pour la transaction.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui renvoie l'horodatage de date de fin de l'année d'une transaction est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`

Pour calculer l'année d'une transaction, créez les mesures suivantes :

- `=yearend(date)`
- `=timestamp(yearend(date))`

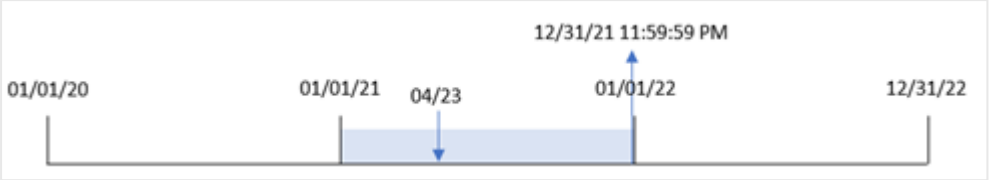
Tableau de résultats

id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

La mesure '`end_of_year`' est créée dans l'objet graphique via la fonction `yearend()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearend()` identifie initialement l'année dans laquelle tombe la valeur date en renvoyant un horodatage pour la dernière milliseconde de cette année.

Diagramme de la fonction `yearend()` montrant que la transaction 8199 a eu lieu en avril.



La transaction 8199 a lieu le 23 avril 2021. La fonction `yearend()` renvoie la dernière milliseconde de cette année, soit le 31 décembre à 11:59:59 PM.

Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Employee_Expenses`'. La table contient les champs suivants :
    - ID des employés
    - Nom de l'employé
    - Notes de frais quotidiennes moyennes de chaque employé

L'utilisateur final souhaite un objet graphique qui affiche, par ID d'employé et nom d'employé, les notes de frais estimées qu'il reste à encourir pour le reste de l'année. L'exercice financier commence en janvier.

Script de chargement

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- employee_id
- employee_name

Pour calculer les notes de frais projetées, créez la mesure suivante :

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

employee_id	employee_name	=(yearend(today(1))-today(1))*avg_daily_claim
182	Mark	$3240.00
183	Deryck	$2700.00
184	Dexter	$2700.00
185	Sydney	$5832.00
186	Agatha	$3888.00

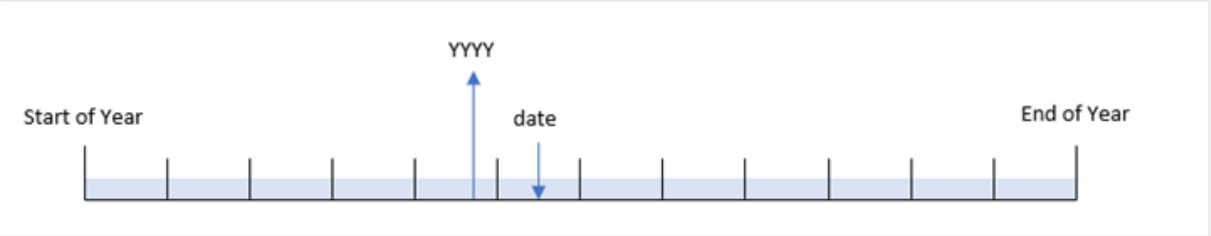
Si on utilise la date d'aujourd'hui comme seul argument, la fonction `yearend()` renvoie la date de fin de l'année en cours. Ensuite, si on soustraie la date d'aujourd'hui de la date de fin d'année, l'expression renvoie le nombre de jours restants pour cette année.

Cette valeur est ensuite multipliée par les notes de frais quotidiennes moyennes par employé pour calculer la valeur estimée des notes de frais que chaque employé est censé faire au cours de l'année restante.

yearname

Cette fonction renvoie une année composée de quatre chiffres comme valeur d'affichage avec une valeur numérique sous-jacente correspondant à un horodatage de la première milliseconde du premier jour de l'année contenant l'argument **date**.

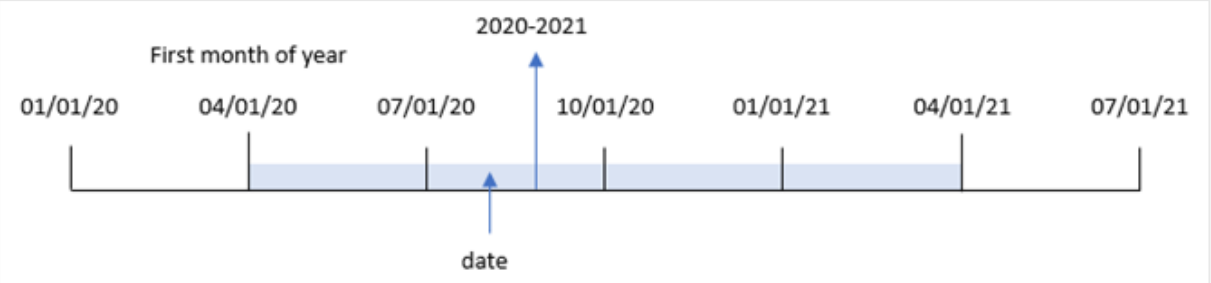
Diagramme de plage de temps de la fonction `yearname()`.



La fonction `yearname()` est différente de la fonction `year()`, car elle vous permet de décaler la date que vous souhaitez évaluer et de définir le premier mois de l'année.

Si le premier mois de l'année n'est pas janvier, la fonction renverra les deux années de quatre chiffres sur la période de douze mois contenant la date. Par exemple, si le début de l'année est avril et si la date évaluée est 06/30/2020, le résultat renvoyé sera `2020-2021`.

Diagramme de la fonction `yearname()` avec avril défini comme le premier mois de l'année.



Syntaxe :

```
YearName(date[, period_no[, first_month_of_year]] )
```

Type de données renvoyé : double

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique l'année comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les années passés tandis que les valeurs positives désignent les années à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**. La valeur d'affichage sera alors une chaîne indiquant deux années.

Vous pouvez utiliser les valeurs suivantes pour définir le premier mois de l'année dans l'argument `first_month_of_year` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Cas d'utilisation

La fonction `yearname()` est utile pour comparer des agrégations par année. Par exemple, si vous souhaitez voir les ventes totales de produits par an.

Il est possible de créer ces dimensions dans le script de chargement via la fonction permettant de créer un champ dans une table Calendrier principal. Elles peuvent également être créées dans un graphique sous forme de dimensions calculées.

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`yearname('10/19/2001')`	Renvoie '2001.'
`yearname('10/19/2001',-1)`	Renvoie '2000.'
`yearname('10/19/2001',0,4)`	Renvoie '2001-2002.'

Rubriques connexes

Rubrique	Description
year (page 1130)	Cette fonction renvoie un entier représentant l'année au cours de laquelle l'expression est interprétée comme une date selon l'interprétation standard des nombres.

Exemple 1 – aucun argument supplémentaire

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Variable système `DateFormat` définie sur '`MM/DD/YYYY`'.
- Chargement précédent utilisant la valeur `yearname()` et défini sous forme de champ `year_name`.

Script de chargement

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

Transactions:
    Load
      *,
      yearname(date) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `year_name`

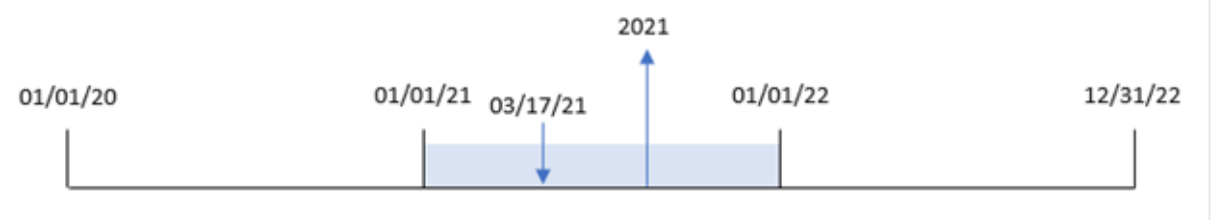
Tableau de résultats

date	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Le champ '`year_name`' est créé dans l'instruction LOAD précédente via la fonction `yearname()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearname()` identifie l'année dans laquelle tombe la valeur de date et la renvoie sous la forme d'une valeur d'année à quatre chiffres.

Diagramme de la fonction `yearname()` qui affiche 2021 comme valeur de l'année.



Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée 'Transactions'.
- Variable système DateFormat définie sur '`MM/DD/YYYY`'.
- Chargement précédent utilisant la valeur `yearname()` et défini sous forme de champ `year_name`.

Script de chargement

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

Transactions:
    Load
      *,
      yearname(date,-1) as prior_year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

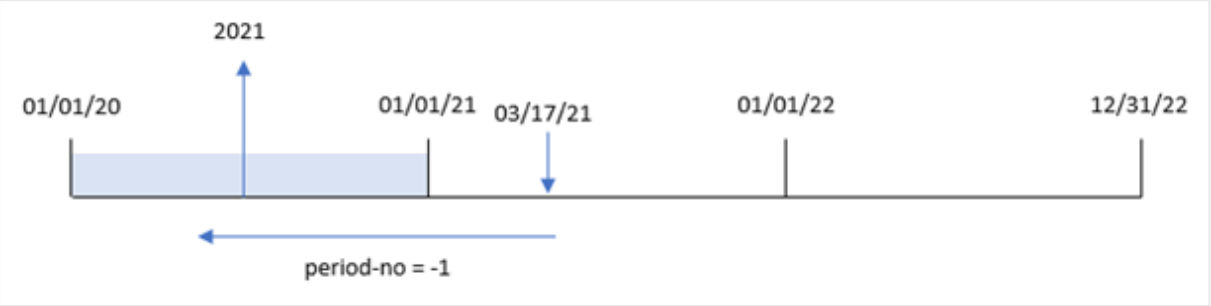
- `date`
- `prior_year_name`

Tableau de résultats

date	prior_year_name
01/13/2020	2019
02/26/2020	2019
03/27/2020	2019
04/16/2020	2019
05/21/2020	2019
08/14/2020	2019
10/07/2020	2019
12/05/2020	2019
01/22/2021	2020
02/03/2021	2020
03/17/2021	2020
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

Étant donné qu'une valeur `period_no` de `-1` est utilisée comme argument de décalage dans la fonction `yearname()`, la fonction commence par identifier l'année au cours de laquelle les transactions ont lieu. La fonction se décale ensuite d'un an en arrière et renvoie l'année obtenue.

Diagramme de la fonction `yearname()` avec la valeur `period_no` définie sur `-1`.



Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- Variable système `DateFormat` définie sur '`MM/DD/YYYY`'.
- Chargement précédent utilisant la valeur `yearname()` et défini sous forme de champ `year_name`.

Script de chargement

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

Transactions:
    Load
      *,
      yearname(date,0,4) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
```

```
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `year_name`

Tableau de résultats

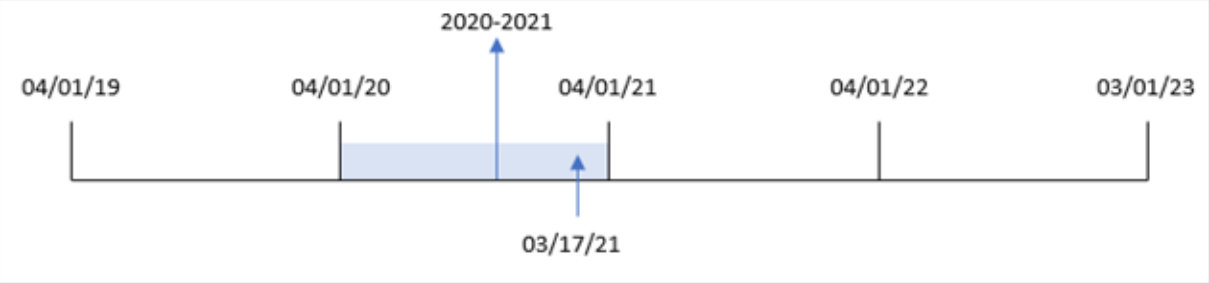
date	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021
01/22/2021	2020-2021
02/03/2021	2020-2021
03/17/2021	2020-2021
04/23/2021	2021-2022
05/04/2021	2021-2022
06/30/2021	2021-2022
07/26/2021	2021-2022
12/27/2021	2021-2022
06/06/2022	2022-2023
07/18/2022	2022-2023

date	year_name
11/14/2022	2022-2023
12/12/2022	2022-2023

Étant donné que l'argument `first_month_of_year` de `4` est utilisé dans la fonction `yearname()`, le début de l'année se déplace du 1er janvier au 1er avril. Par conséquent, chaque période de douze mois traverse deux années civiles et la fonction `yearname()` renvoie les deux années à quatre chiffres des dates évaluées.

La transaction 8198 a lieu le 17 mars 2021. La fonction `yearname()` définit le début de l'année comme le 1er avril et la fin comme le 30 mars. Par conséquent, la transaction 8198 a eu lieu au cours de la période annuelle du 1er avril 2020 au 30 mars 2021. C'est pourquoi la fonction `yearname()` renvoie la valeur 2020-2021.

Diagramme de la fonction `yearname()` avec mars défini comme le premier mois de l'année.



Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- Variable système `DateFormat` définie sur '`MM/DD/YYYY`'.

Cependant, le champ qui renvoie l'année au cours de laquelle la transaction a eu lieu est créé sous forme de mesure dans un objet graphique.

Script de chargement

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

Transactions:
Load
*
```

```
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension :

`date`

Pour calculer le champ '`year_name`', créez la mesure suivante :

```
=yearname(date)
```

Tableau de résultats

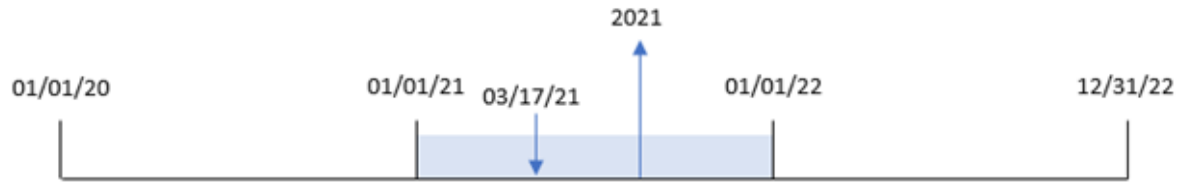
date	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021

date	=yearname(date)
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

La mesure '`year_name`' est créée dans l'objet graphique via la fonction `yearname()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearname()` identifie l'année dans laquelle tombe la valeur de date et la renvoie sous la forme d'une valeur d'année à quatre chiffres.

Diagramme de la fonction `yearname()` avec 2021 comme valeur d'année.



Exemple 5 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Le même ensemble de données que dans le premier exemple.
- Variable système `DateFormat` définie sur '`MM/DD/YYYY`'.

L'utilisateur final souhaite un graphique présentant les ventes totales par trimestre pour les transactions. Utilisez la fonction `yearname()` comme dimension calculée pour créer ce graphique lorsque la dimension `yearname()` n'est pas disponible dans le modèle de données.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez un tableau.

Pour comparer des agrégations par année, créez cette dimension calculée :

```
=yearname(date)
```

Créez cette mesure :

```
=sum(amount)
```

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

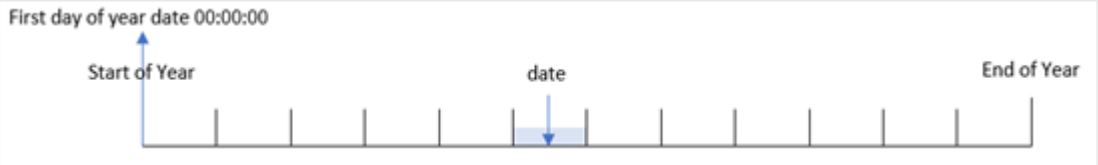
Cette fonction renvoie un horodatage correspondant au début du premier jour de l'année contenant l'argument **date**. Le format de sortie par défaut correspond à l'argument **DateFormat** défini dans le script.

Syntaxe :

```
YearStart(date[, period_no[, first_month_of_year]])
```

En d'autres termes, la fonction `yearstart()` détermine l'année dans laquelle tombe la date. Elle renvoie ensuite un horodatage, au format date, pour la première milliseconde de cette année. Le premier mois de l'année est, par défaut, le mois de janvier ; cependant, vous pouvez modifier le mois défini comme le premier en utilisant l'argument `first_month_of_year` dans la fonction `yearstart()`.

Diagramme de la fonction `yearstart()` montrant la plage de temps que la fonction peut couvrir.



Cas d'utilisation

La fonction `yearstart()` est utilisée dans le cadre d'une expression lorsque vous souhaitez que le calcul utilise la fraction de l'année qui s'est écoulée jusqu'ici. Par exemple, si vous souhaitez calculer les intérêts accumulés au cours d'une année jusqu'à ce jour.

Type de données renvoyé : double

Arguments

Argument	Description
date	Date ou horodatage à évaluer.
period_no	**period_no** est un entier, où la valeur 0 indique l'année comprenant l'argument **date**. Les valeurs négatives de l'argument **period_no** indiquent les années passés tandis que les valeurs positives désignent les années à venir.
first_month_of_year	Si vous voulez utiliser des exercices (financiers) qui ne commencent pas en janvier, indiquez une valeur comprise entre 2 et 12 dans l'argument **first_month_of_year**.

Les mois suivants peuvent être utilisés dans l'`first_month_of_year argument` :

Valeurs first_month_of_year

Mois	Valeur
Février	2
Mars	3
Avril	4
Mai	5
Juin	6
Juillet	7
Août	8
Septembre	9
Octobre	10
Novembre	11
Décembre	12

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemples de fonction

Exemple	Résultat
`yearstart('10/19/2001')`	Renvoie 01/01/2001 00:00:00.
`yearstart('10/19/2001',-1)`	Renvoie 01/01/2000 00:00:00.
`yearstart('10/19/2001',0,4)`	Renvoie 04/01/2001 00:00:00.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée '`Transactions`'.
- Champ de date fourni dans la variable système `DateFormat` au format (`MM/DD/YYYY`).
- Instruction LOAD précédente contenant les éléments suivants :
  - La fonction `yearstart()` définie comme le champ `year_start`.
  - La fonction `Timestamp()` définie comme le champ `year_start_timestamp`

Script de chargement

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

Transactions:
    Load
       *,
       yearstart(date) as year_start,
       timestamp(yearstart(date)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
```

```
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Tableau de résultats

id	date	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8204	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8205	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8206	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8207	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM

Le champ '`year_start`' est créé dans l'instruction LOAD précédente via la fonction `yearstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearstart()` identifie initialement l'année dans laquelle tombe la valeur date et renvoie un horodatage pour la première milliseconde de cette année-là.

Diagramme de la fonction `yearstart()` et de la transaction 8199.

La transaction 8199 a eu lieu le 23 avril 2021. La fonction `yearstart()` renvoie la première milliseconde de cette année-là, soit le 1er janvier à 12:00:00 AM.

Exemple 2 – period_no

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la tâche consiste à créer un champ, '`previous_year_start`', qui renvoie l'horodatage de la date de début de l'année précédant l'année où la transaction a eu lieu.

Script de chargement

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

Transactions:
    Load
       *,
       yearstart(date,-1) as previous_year_start,
       timestamp(yearstart(date,-1)) as previous_year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
```

```
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- previous_year_start
- previous_ year_start_timestamp

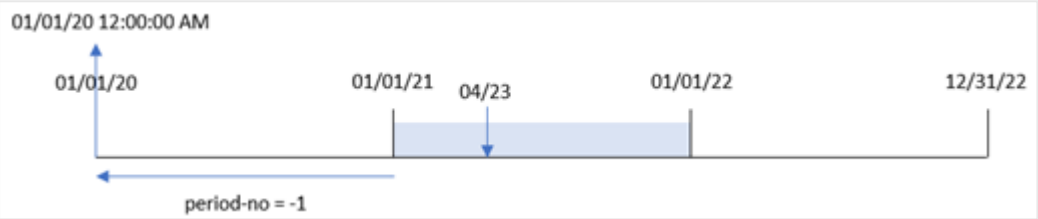
Tableau de résultats

id	date	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM
8189	02/26/2020	01/01/2019	1/1/2019 12:00:00 AM
8190	03/27/2020	01/01/2019	1/1/2019 12:00:00 AM
8191	04/16/2020	01/01/2019	1/1/2019 12:00:00 AM
8192	05/21/2020	01/01/2019	1/1/2019 12:00:00 AM
8193	08/14/2020	01/01/2019	1/1/2019 12:00:00 AM
8194	10/07/2020	01/01/2019	1/1/2019 12:00:00 AM
8195	12/05/2020	01/01/2019	1/1/2019 12:00:00 AM
8196	01/22/2021	01/01/2020	1/1/2020 12:00:00 AM
8197	02/03/2021	01/01/2020	1/1/2020 12:00:00 AM
8198	03/17/2021	01/01/2020	1/1/2020 12:00:00 AM
8199	04/23/2021	01/01/2020	1/1/2020 12:00:00 AM
8200	05/04/2021	01/01/2020	1/1/2020 12:00:00 AM
8201	06/30/2021	01/01/2020	1/1/2020 12:00:00 AM
8202	07/26/2021	01/01/2020	1/1/2020 12:00:00 AM
8203	12/27/2021	01/01/2020	1/1/2020 12:00:00 AM
8204	06/06/2022	01/01/2021	1/1/2021 12:00:00 AM
8205	07/18/2022	01/01/2021	1/1/2021 12:00:00 AM

id	date	previous_year_start	previous_year_start_timestamp
8206	11/14/2022	01/01/2021	1/1/2021 12:00:00 AM
8207	12/12/2022	01/01/2021	1/1/2021 12:00:00 AM

Dans cette instance, étant donné qu'une valeur `period_no` de `-1` est utilisée comme argument de décalage dans la fonction `yearstart()`, la fonction commence par identifier l'année au cours de laquelle les transactions ont lieu. Elle regarde ensuite un an avant et identifie la première milliseconde de cette année-là.

Diagramme de la fonction `yearstart()` avec une valeur `period_no` de -1.



La transaction 8199 a eu lieu le 23 avril 2021. La fonction `yearstart()` renvoie la première milliseconde de l'année précédente, soit le 1er janvier 2020 à 12:00:00 AM, pour le champ '`previous_year_start`'.

Exemple 3 – first_month_of_year

Script de chargement et résultats

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, la politique de l'entreprise stipule que l'année commence le 1er avril.

Script de chargement

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

Transactions:
    Load
       *,
       yearstart(date,0,4) as year_start,
       timestamp(yearstart(date,0,4)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
```

```
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- id
- date
- year_start
- year_start_timestamp

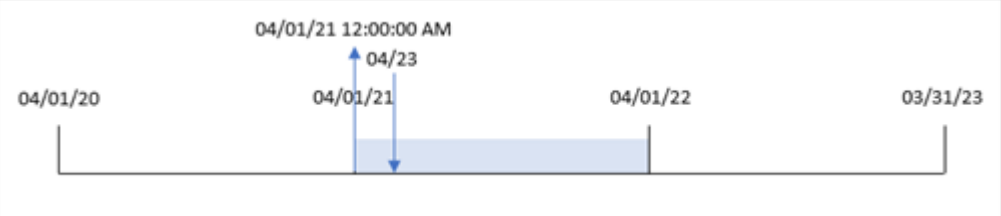
Tableau de résultats

id	date	year_start	year_start_timestamp
8188	01/13/2020	04/01/2019	4/1/2019 12:00:00 AM
8189	02/26/2020	04/01/2019	4/1/2019 12:00:00 AM
8190	03/27/2020	04/01/2019	4/1/2019 12:00:00 AM
8191	04/16/2020	04/01/2020	4/1/2020 12:00:00 AM
8192	05/21/2020	04/01/2020	4/1/2020 12:00:00 AM
8193	08/14/2020	04/01/2020	4/1/2020 12:00:00 AM
8194	10/07/2020	04/01/2020	4/1/2020 12:00:00 AM
8195	12/05/2020	04/01/2020	4/1/2020 12:00:00 AM
8196	01/22/2021	04/01/2020	4/1/2020 12:00:00 AM
8197	02/03/2021	04/01/2020	4/1/2020 12:00:00 AM
8198	03/17/2021	04/01/2020	4/1/2020 12:00:00 AM
8199	04/23/2021	04/01/2021	4/1/2021 12:00:00 AM
8200	05/04/2021	04/01/2021	4/1/2021 12:00:00 AM

id	date	year_start	year_start_timestamp
8201	06/30/2021	04/01/2021	4/1/2021 12:00:00 AM
8202	07/26/2021	04/01/2021	4/1/2021 12:00:00 AM
8203	12/27/2021	04/01/2021	4/1/2021 12:00:00 AM
8204	06/06/2022	04/01/2022	4/1/2022 12:00:00 AM
8205	07/18/2022	04/01/2022	4/1/2022 12:00:00 AM
8206	11/14/2022	04/01/2022	4/1/2022 12:00:00 AM
8207	12/12/2022	04/01/2022	4/1/2022 12:00:00 AM

Dans cette instance, étant donné que l'argument `first_month_of_year` égal à `4` est utilisé dans la fonction `yearstart()`, le premier jour de l'année est défini sur le 1er avril et le dernier jour de l'année sur le 31 mars.

Diagramme de la fonction `yearstart()` avec le premier mois défini sur avril.



La transaction 8199 a eu lieu le 23 avril 2021. Parce que la fonction `yearstart()` définit le début de l'année sur le 1er avril et le renvoie comme la valeur '`year_start`' pour la transaction.

Exemple 4 – Exemple d'objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Le même ensemble de données et le même scénario que ceux du premier exemple sont utilisés.

Cependant, dans cet exemple, l'ensemble de données est le même et chargé dans l'application. Le calcul qui renvoie l'horodatage de date de début de l'année de réalisation d'une transaction est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
```

```
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `id`
- `date`

Pour calculer l'année d'une transaction, créez les mesures suivantes :

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

Tableau de résultats

id	date	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8189	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8190	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8191	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM
8192	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8193	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8194	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8195	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8196	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM

id	date	=yearstart(date)	=timestamp(yearstart(date))
8199	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8201	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8202	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8203	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8204	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8205	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8206	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8207	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM

La mesure '`start_of_year`' est créée dans l'objet graphique via la fonction `yearstart()` et en transmettant le champ date comme argument de la fonction.

La fonction `yearstart()` identifie initialement l'année dans laquelle tombe la valeur date et renvoie un horodatage pour la première milliseconde de cette année-là.

Diagramme de la fonction `yearstart()` et de la transaction 8199.

La transaction 8199 a eu lieu le 23 avril 2021. La fonction `yearstart()` renvoie la première milliseconde de cette année-là, soit le 1er janvier à 12:00:00 AM.

Exemple 5 – scénario

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données chargé dans une table appelée '`Loans`'. La table contient les champs suivants :
  - Les ID de prêt.

- Le solde en début d'année.
- Le taux d'intérêt simple facturé pour chaque prêt par an.

L'utilisateur final souhaite un objet graphique qui affiche, par ID de prêt, les intérêts actuels qui ont été accumulés pour chaque prêt au cours de l'année jusqu'à la date du jour.

Script de chargement

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `loan_id`
- `start_balance`

Pour calculer les intérêts accumulés, créez la mesure suivante :

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Définissez le **Formatage des nombres** des mesures sur **Money** (Devise).

Tableau de résultats

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8188	$10000.00	$39.73
8189	$15000.00	$339.66
8190	$17500.00	$166.85
8191	$21000.00	$283.64
8192	$90000.00	$3003.29

Si on utilise la date d'aujourd'hui comme seul argument, la fonction `yearstart()` renvoie la date de début de l'année en cours. Si on soustraie ce résultat de la date actuelle, l'expression renvoie le nombre de jours qui se sont écoulés jusqu'à présent cette année.

Cette valeur est ensuite multipliée par le taux d'intérêt et divisée par 365 pour obtenir le taux d'intérêt effectif pour la période. Le taux d'intérêt effectif pour la période est ensuite multiplié par le solde initial du prêt pour renvoyer les intérêts accumulés jusqu'à présent cette année.

yeartodate

Cette fonction permet de déterminer si l'horodatage tombe dans l'année de la date à laquelle le script a été chargé pour la dernière fois et renvoie True si c'est le cas ou False si ce n'est pas le cas.

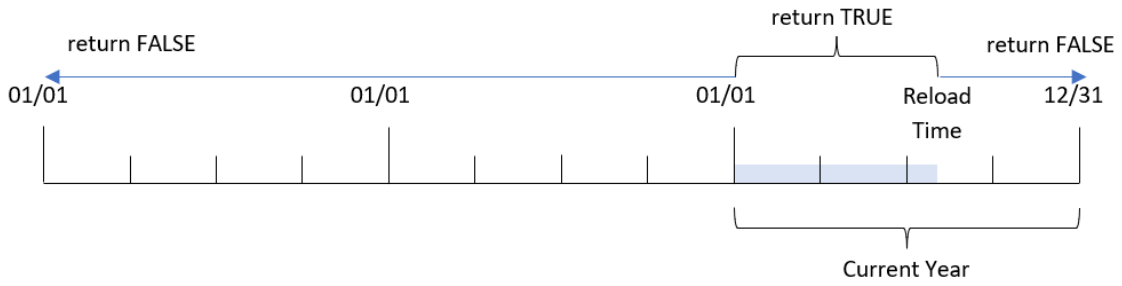
Syntaxe :

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Type de données renvoyé : booléen

> *Dans Qlik Sense, la valeur booléenne true est représentée par -1 et la valeur false par 0.*

Exemple de diagramme de la fonction `yeartodate()`



Si aucun des paramètres facultatifs n'est utilisé, l'année en cours correspond à toute date comprise dans une année calendaire, qui s'étend du 1er janvier à la date de la dernière exécution du script comprise.

En d'autres termes, la fonction `yeartodate()`, lorsqu'elle est déclenchée sans aucun paramètre supplémentaire, est utilisée pour évaluer un horodatage et renvoyer un résultat booléen selon que la date est tombée pendant l'année civile jusqu'à la date d'actualisation incluse.

Cependant, il est également possible de remplacer la date de début de l'année via l'argument `firstmonth`, ainsi que d'effectuer des comparaisons avec des années précédentes ou suivantes via l'argument `yearoffset`.

Pour finir, dans les cas d'ensembles de données historiques, la fonction `yeartodate()` fournit un paramètre pour définir `todaydate`, qui, à la place, comparera l'horodatage à l'année civile jusqu'à la date incluse fournie dans l'argument `todaydate`.

Arguments

Argument	Description
timestamp	Horodatage à évaluer, par exemple '10/12/2012'.

Argument	Description
yearoffset	En spécifiant une valeur pour **yearoffset**, **yeartodate** renvoie True pour la même période d'une autre année. Un décalage **yearoffset** négatif indique une année antérieure tandis qu'un décalage positif indique une année ultérieure. La valeur year-to-date la plus récente est obtenue en spécifiant yearoffset = -1. Si cette valeur est omise, 0 est défini par défaut.
firstmonth	Si on spécifie un argument **firstmonth** compris entre 1 et 12 (1 si l'argument est omis), il est possible de déplacer le début de l'année au premier jour de n'importe quel mois. Par exemple, si vous voulez travailler sur un exercice fiscal débutant le premier mai, spécifiez **firstmonth** = 5. Une valeur 1 indiquerait un exercice financier commençant le 1er janvier et une valeur 12 un exercice financier commençant le 1er décembre.
todaydate	Vous pouvez déplacer le jour utilisé comme limite supérieure de la période en indiquant une date **todaydate** (horodatage de la dernière exécution du script si l'argument est omis).

Cas d'utilisation

La fonction `yeartodate()` renvoie un résultat booléen. Ce type de fonction sera généralement utilisé comme condition dans une expression if. Cela renverrait une agrégation ou un calcul suivant que la date évaluée s'est produite ou non au cours de l'année jusqu'à la date, incluse, du dernier chargement de l'application.

Par exemple, la fonction YearToDate() peut être utilisée pour identifier l'ensemble des équipements fabriqués jusqu'à présent au cours de l'année en cours.

Dans les exemples suivants, nous supposons que la date du dernier chargement = 11/18/2011.

Exemples de fonction

Exemple	Résultat
`yeartodate( '11/18/2010')`	renvoie False
`yeartodate( '02/01/2011')`	renvoie True
`yeartodate( '11/18/2011')`	renvoie True
`yeartodate( '11/19/2011')`	renvoie False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	renvoie True
`yeartodate( '11/18/2010', -1)`	renvoie True
`yeartodate( '11/18/2011', -1)`	renvoie False
`yeartodate( '04/30/2011', 0, 5)`	renvoie False
`yeartodate( '05/01/2011', 0, 5)`	renvoie True

Paramètres régionaux

Sauf indication contraire, les exemples de cette rubrique utilisent le format de date suivant : MM/JJ/AAAA. Le format de date est indiqué dans l'instruction `SET DateFormat` de votre script de chargement de données. Le format de date par défaut peut être différent dans votre système en raison de vos paramètres régionaux et d'autres facteurs. Vous pouvez modifier les formats utilisés dans les exemples ci-dessous en fonction de vos besoins. Ou vous pouvez modifier les formats utilisés dans votre script de chargement pour qu'ils correspondent à ceux de ces exemples.

Les paramètres régionaux par défaut des applications sont basés sur les paramètres système régionaux de l'ordinateur ou du serveur sur lequel Qlik Sense est installé. Si le serveur Qlik Sense auquel vous accédez est configuré sur la Suède, l'éditeur de chargement de données utilisera les paramètres régionaux suédois pour les dates, l'heure et la devise. Ces paramètres de format régionaux ne sont pas liés à la langue affichée dans l'interface utilisateur Qlik Sense. Qlik Sense sera affiché dans la même langue que celle du navigateur que vous utilisez.

Exemple 1 – exemple de base

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).
- Création d'un champ, `year_to_date`, qui détermine les transactions qui ont eu lieu au cours de l'année civile jusqu'à la date du dernier chargement.

La date du jour est le 26 avril 2022.

Script de chargement

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

Transactions:
    Load
        *,
        yeartodate(date) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
```

```
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

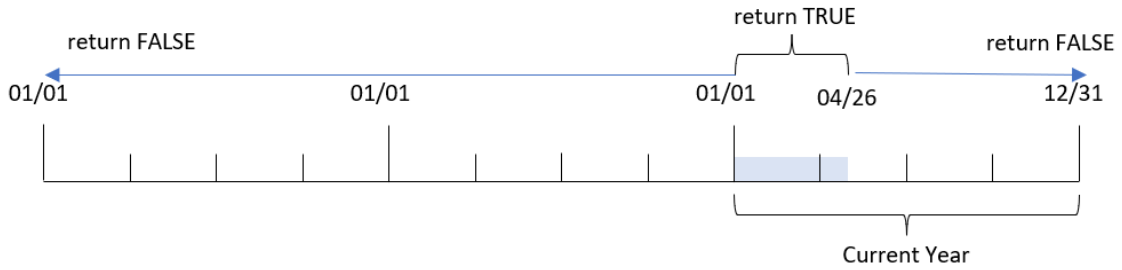
- `date`
- `year_to_date`

Tableau de résultats

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0

date	year_to_date
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagramme de la fonction `yeartodate()`, exemple de base



Le champ `year_to_date` est créé dans l'instruction LOAD précédente via la fonction `yeartodate()` et en transmettant le champ `date` comme argument de la fonction.

Étant donné qu'aucun autre paramètre n'est transmis à la fonction, la fonction `yeartodate()` commence par identifier la date de chargement et par conséquent les limites de l'année civile en cours (commençant le 1er janvier) qui renverront un résultat booléen `TRUE`.

Par conséquent, toute transaction effectuée entre le 1er janvier et le 26 avril, la date d'actualisation renverra un résultat booléen `TRUE`. Toute transaction qui a lieu avant le début de l'année 2022 renverra un résultat booléen `FALSE`.

Exemple 2 – yearoffset

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `two_years_prior`, qui détermine les transactions qui ont eu lieu deux années complètes avant l'année civile jusqu'à ce jour.

Script de chargement

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

Transactions:
    Load
        *,
        yeartodate(date,-2) as two_years_prior
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `two_years_prior`

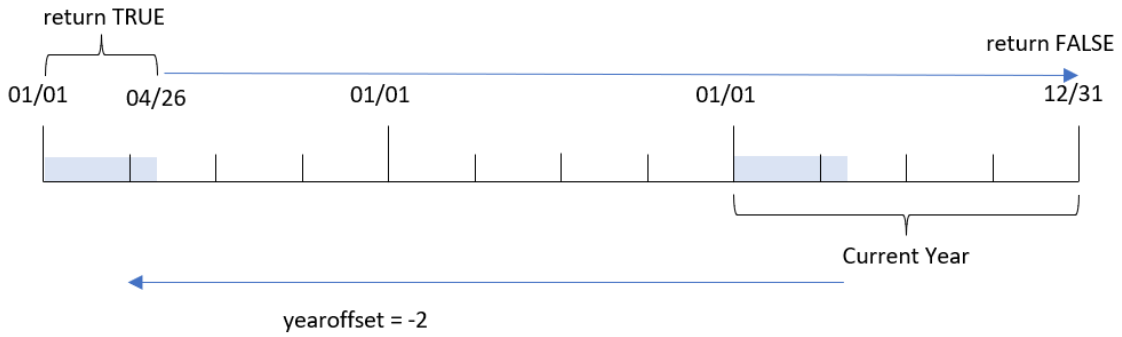
Tableau de résultats

date	two_years_prior
01/10/2020	-1

date	two_years_prior
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	0
02/26/2022	0
03/07/2022	0
03/11/2022	0

Avec l'utilisation de la valeur -2 comme argument `yearoffset` dans la fonction `yeartodate()`, la fonction décale les limites du segment d'année civile de comparaison de deux années complètes. Initialement, le segment d'année est compris entre le 1er janvier et le 26 avril 2022. L'argument `yearoffset` décale ensuite ce segment de deux années en arrière. Les limites de date deviennent alors du 1er janvier au 26 avril 2020.

Diagramme de la fonction `yeartodate()`, exemple yearoffset



Par conséquent, toute transaction effectuée entre le 1er janvier et le 26 avril 2020 renverra un résultat booléen TRUE. Toute transaction qui apparaît avant ou après ce segment renverra FALSE.

Exemple 3 – firstmonth

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `year_to_date`, qui détermine les transactions qui ont eu lieu au cours de l'année civile jusqu'à la date du dernier chargement.

Dans cet exemple, nous déterminons le début de l'exercice fiscal au 1er juillet.

Script de chargement

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

Transactions:
    Load
        *,
        yeartodate(date,0,7) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
```

```
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `year_to_date`

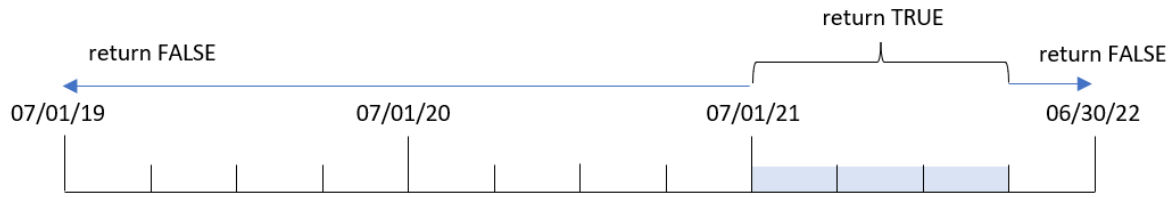
Tableau de résultats

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0

date	year_to_date
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Dans ce cas, étant donné que l'argument `firstmonth` égal à 7 est utilisé dans la fonction `yeartodate()`, le premier jour de l'année est défini sur le 1er juillet et le dernier jour de l'année sur le 30 juin.

Diagramme de la fonction `yeartodate()`, exemple firstmonth



Par conséquent, toute transaction effectuée entre le 1er juillet 2021 et le 26 avril 2022, la date d'actualisation, renverra un résultat booléen `TRUE`. Toute transaction qui a lieu avant le 1er juillet 2021 renverra un résultat booléen `FALSE`.

Exemple 4 – todaydate

Script de chargement et résultats

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Même ensemble de données et même scénario que ceux du premier exemple.
- Création d'un champ, `year_to_date`, qui détermine les transactions qui ont eu lieu au cours de l'année civile jusqu'à la date du dernier chargement.

Cependant, dans cet exemple, nous devons identifier toutes les transactions qui ont eu lieu au cours de l'année civile jusqu'à la date incluse du 1er mars 2022.

Script de chargement

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

Transactions:
    Load
        *,
        yeartodate(date, 0, 1, '03/01/2022') as year_to_date
;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ces champs comme dimensions :

- `date`
- `year_to_date`

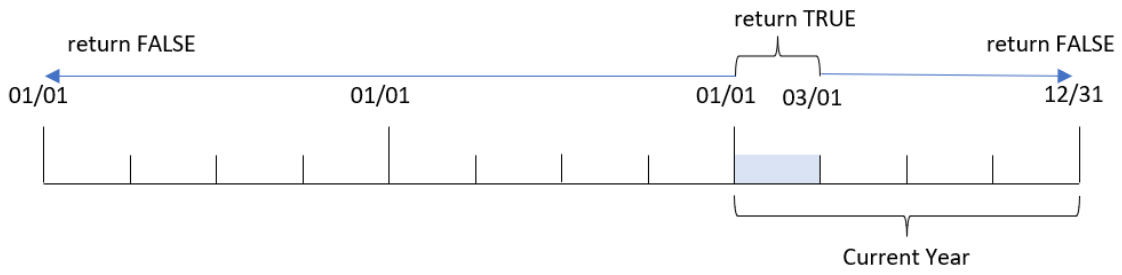
Tableau de résultats

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0

date	year_to_date
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	0
03/11/2022	0

Dans ce cas, étant donné que l'argument `todaydate` de 03/01/2022 est utilisé dans la fonction `yeartodate ()`, il détermine la limite de fin du segment d'année civile de comparaison au 1er mars 2022. Il est essentiel de fournir le paramètre `firstmonth` (compris entre 1 et 12) ; sinon, la fonction renverra des résultats nuls.

Diagramme de la fonction `yeartodate()`, exemple avec l'argument todaydate



Par conséquent, toute transaction effectuée entre le 1er janvier 2022 et le 1er mars 2022, le paramètre `todaydate`, renverra un résultat booléen `TRUE`. Toute transaction qui a lieu avant le 1er janvier 2022 ou après le 1er mars 2022 renverra un résultat booléen `FALSE`.

Exemple 5 – exemple objet graphique

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient le même ensemble de données et le même scénario que ceux du premier exemple.

Cependant, dans cet exemple, le même ensemble de données est chargé dans l'application. Le calcul qui détermine les transactions qui ont eu lieu au cours de l'année civile jusqu'à la date du dernier chargement est créé sous forme de mesure dans un objet graphique de l'application.

Script de chargement

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Chargez les données et ouvrez une feuille. Créez une table et ajoutez ce champ comme dimension : `date`.

Ajoutez la mesure suivante :

```
=yeartodate(date)
```

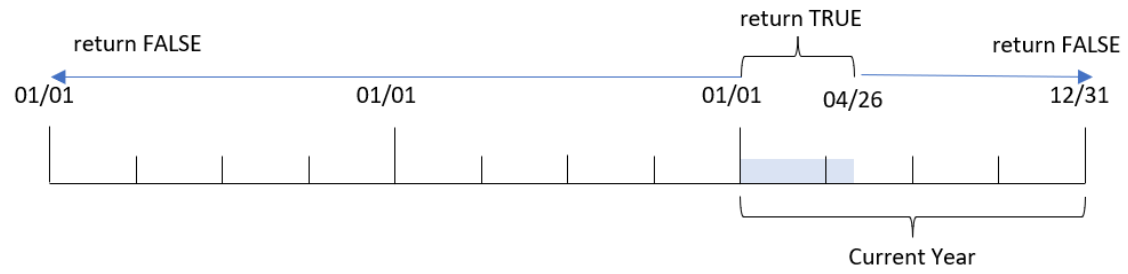
Tableau de résultats

date	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

La mesure `year_to_date` est créée dans l'objet graphique via la fonction `yeartodate()` et en transmettant le champ `date` comme argument de la fonction.

Étant donné qu'aucun autre paramètre n'est transmis à la fonction, la fonction `yeartodate()` commence par identifier la date de chargement et par conséquent les limites de l'année civile en cours (commençant le 1er janvier) qui renverront un résultat booléen `TRUE`.

Diagramme de la fonction `yeartodate()`, exemple avec un objet graphique



Toute transaction effectuée entre le 1er janvier et le 26 avril, la date d'actualisation, renverra un résultat booléen `TRUE`. Toute transaction qui a lieu avant le début de l'année 2022 renverra un résultat booléen `FALSE`.

Exemple 6 – scénario

Script de chargement et expression de graphique

Vue d'ensemble

Ouvrez l'éditeur de chargement de données et ajoutez le script de chargement ci-dessous à un nouvel onglet.

Le script de chargement contient :

- Ensemble de données contenant un ensemble de transactions entre 2020 et 2022, chargé dans une table appelée `Transactions`.
- Champ de date fourni dans la variable système `DateFormat` au format (MM/DD/YYYY).

L'utilisateur final souhaite un objet Indicateur KPI présentant les ventes totales, pour la période équivalente en 2021, sous la forme d'une analyse Même période d'une année à l'autre, à la date du dernier chargement.

La date du jour est le 16 juin 2022.

Script de chargement

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

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
```

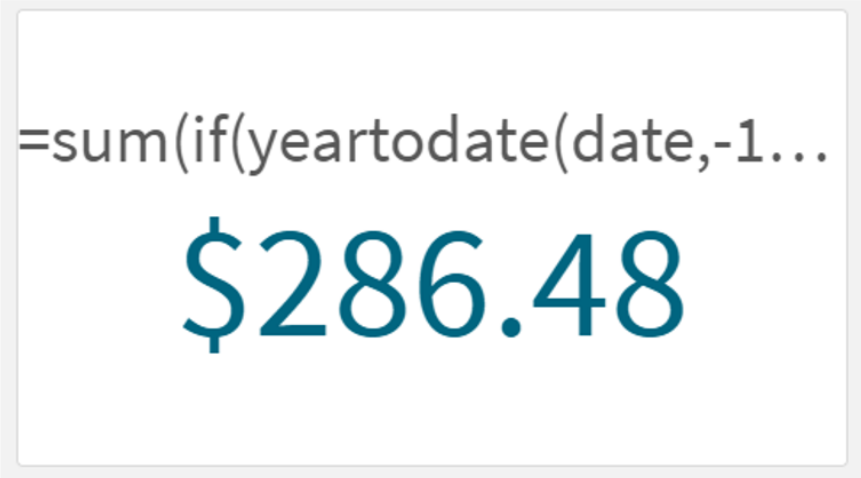
```
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Résultats

Procédez comme suit :

1. Créez un objet KPI.
2. Pour calculer les ventes totales, créez la mesure d'agrégation suivante : `=sum(if(yeartodate(date,-1),amount,0))`
3. Définissez le **Formatage des nombres** de la mesure sur **Devise**.

Graphique Indicateur KPI `yeartodate()` pour 2021



La fonction `yeartodate()` renvoie une valeur booléenne lors de l'évaluation des dates de chaque ID de transaction. Étant donné que le chargement a eu lieu le 16 juin 2022, la fonction `yeartodate` segmente la période year du 01/01/2022 au 06/16/2022. Cependant, étant donné qu'une valeur `period_no` égale à -1 a

été utilisée dans la fonction, ces limites sont décalées à l'année précédente. Par conséquent, pour toute transaction qui se produit entre le 01/01/2021 et le 06/16/2021, la fonction `yeartodate()` renvoie une valeur booléenne `TRUE` et additionne le montant.

5.8 Fonctions exponentielles et logarithmiques

Cette section décrit les fonctions relatives aux calculs exponentiels et logarithmiques. Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

Dans les fonctions ci-dessous, les paramètres sont des expressions dans lesquelles **x** et **y** doivent être interprétés comme des nombres réels.

exp

Fonction exponentielle népérienne ou naturelle, e^x, utilisant le logarithme népérien **e** comme base. Le résultat est un nombre positif.

```
exp(x )
```

Exemples et résultats :

`exp(3)` renvoie 20.085.

log

Logarithme népérien de **x**. La fonction est uniquement définie si **x**> 0. Le résultat est un nombre.

```
log(x )
```

Exemples et résultats :

`log(3)` renvoie 1.0986.

log10

Logarithme décimal (de base 10) de **x**. La fonction est uniquement définie si **x**> 0. Le résultat est un nombre.

```
log10(x )
```

Exemples et résultats :

`log10(3)` renvoie 0.4771.

pow

Renvoie **x** à la puissance **y**. Le résultat est un nombre.

```
pow(x,y )
```

Exemples et résultats :

`pow(3, 3)` renvoie 27.

sqr
x au carré (**x** à la puissance 2). Le résultat est un nombre.

```
sqr (x )
```

Exemples et résultats :

`sqr(3)` renvoie 9.

sqrt
Racine carrée de **x**. La fonction est uniquement définie si **x** >= 0. Le résultat est un nombre positif.

```
sqrt(x )
```

Exemples et résultats :

`sqrt(3)` renvoie 1.732.

5.9 Fonctions de champ

Ces fonctions s'utilisent uniquement dans les expressions de graphique.

Les fonctions de champ renvoient soit des entiers soit des chaînes permettant d'identifier différents aspects des sélections de champ.

Fonctions de nombre

GetAlternativeCount
GetAlternativeCount() permet de déterminer le nombre de valeurs alternatives (en gris clair) dans le champ identifié.

```
GetAlternativeCount - fonction de graphique (field_name)
```

GetExcludedCount
GetExcludedCount() permet de déterminer le nombre de valeurs exclues dans le champ identifié. Les valeurs exclues incluent les champs alternatifs (en gris clair), exclus (en gris foncé) et sélectionnés exclus (gris foncé avec une coche).

```
GetExcludedCount - fonction de graphique (page 1195)(field_name)
```

GetNotSelectedCount
Cette fonction de graphique renvoie le nombre de valeurs non sélectionnées dans le champ intitulé **fieldname**. Le champ doit être en mode And pour que cette fonction soit pertinente.

```
GetNotSelectedCount - fonction de graphique(fieldname [,
includeexcluded=false])
```

GetPossibleCount

GetPossibleCount() permet de déterminer le nombre de valeurs possibles dans le champ identifié. Si le champ identifié inclut des sélections, les champs sélectionnés (en vert) sont pris en compte. Sinon, ce sont les valeurs associées (en blanc) qui sont comptabilisées.

```
GetPossibleCount - fonction de graphique(field_name)
```

GetSelectedCount

GetSelectedCount() permet de déterminer le nombre de valeurs sélectionnées (en vert) d'un champ.

```
GetSelectedCount - fonction de graphique (field_name [, include_excluded])
```

Fonctions de champ et de sélection

GetCurrentSelections

GetCurrentSelections() renvoie une liste des sélections actives dans l'application. En revanche, si les sélections sont effectuées à l'aide d'une chaîne de recherche indiquée dans la zone de recherche, **GetCurrentSelections()** renvoie la chaîne de recherche.

```
GetCurrentSelections - fonction de graphique([record_sep [,tag_sep [,value_
sep [,max_values]]]])
```

GetFieldSelections

La fonction **GetFieldSelections()** renvoie une chaîne (**string**) avec les sélections actives dans un champ.

```
GetFieldSelections - fonction de graphique ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() renvoie le nom de la dimension. **Index** désigne un entier facultatif qui indique la dimension à renvoyer.

```
GetObjectDimension - fonction de graphique ([index])
```

GetObjectField

GetObjectField() renvoie le nom de la dimension. **Index** désigne un entier facultatif qui indique la dimension à renvoyer.

```
GetObjectField - fonction de graphique ([index])
```

GetObjectMeasure

GetObjectMeasure() renvoie le nom de la mesure. **Index** désigne un entier facultatif qui indique la mesure à renvoyer.

```
GetObjectMeasure - fonction de graphique ([index])
```

GetAlternativeCount - fonction de graphique

GetAlternativeCount() permet de déterminer le nombre de valeurs alternatives (en gris clair) dans le champ identifié.

Syntaxe :

```
GetAlternativeCount (field_name)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
field_name	Champ contenant la plage de données à mesurer.

Exemples et résultats :

Dans l'exemple suivant, le champ **First name** est chargé dans un volet de filtre.

Exemples et résultats

Exemples	Résultats
Supposons que **John** est sélectionné dans **First name**. `GetAlternativeCount ([First name])`	4, car il y a 4 valeurs uniques et exclues (grisées) dans **First name**.
Supposons que **John** et **Peter** sont sélectionnés. `GetAlternativeCount ([First name])`	3, car il y a 3 valeurs uniques et exclues (grisées) dans **First name**.
Supposons qu'aucune valeur n'est sélectionnée dans **First name**. `GetAlternativeCount ([First name])`	0, car aucune sélection n'a été effectuée.

Données utilisées dans l'exemple :

```
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 - fonction de graphique

GetCurrentSelections() renvoie une liste des sélections actives dans l'application. En revanche, si les sélections sont effectuées à l'aide d'une chaîne de recherche indiquée dans la zone de recherche, **GetCurrentSelections()** renvoie la chaîne de recherche.

Si vous utilisez des options, vous devrez spécifier l'argument record_sep. Pour spécifier une nouvelle ligne, définissez **record_sep** sur **chr(13)&chr(10)**.

Si toutes les valeurs sauf deux, ou sauf une, sont sélectionnées, le format 'NOT x,y' ou 'NOT y' sera utilisé. Si vous sélectionnez toutes les valeurs et que le nombre total de valeurs est supérieur à l'argument max_values, le texte ALL est renvoyé.

Syntaxe :

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Arguments	Description
record_sep	Séparateur à placer entre les enregistrements de champ. Le séparateur par défaut est <CR><LF>, qui correspond à l'insertion d'une nouvelle ligne.
tag_sep	Séparateur à placer entre la balise du nom de champ et les valeurs de champ. Par défaut, il s'agit des deux-points « : ».
value_sep	Séparateur à placer entre les valeurs de champ. La valeur par défaut est la virgule (,).
max_values	Nombre maximum de valeurs de champ pouvant être listées individuellement. Lorsqu'un plus grand nombre de valeurs est sélectionné, le format « x valeurs sur y » le remplace. La valeur par défaut est 6.
state_name	Nom d'un état alternatif ayant été choisi pour cette visualisation en particulier. Si l'argument **state_name** est utilisé, seules les sélections associées au nom d'état spécifié sont prises en compte.

Exemples et résultats :

L'exemple suivant utilise deux champs chargés dans des volets de filtre différents, un pour le prénom **First name** et l'autre pour les initiales **Initials**.

Exemples et résultats

Exemples	Résultats
Supposons que **John** est sélectionné dans **First name**. `GetCurrentSelections ()`	'First name: John'
Supposons que **John** et **Peter** sont sélectionnés dans **First name**. `GetCurrentSelections ()`	'First name: John, Peter'

Exemples	Résultats
Supposons que **John** et **Peter** sont sélectionnés dans **First name** et que **JA** est sélectionné dans **Initials**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
Supposons que **John** est sélectionné dans **First name** et que **JA** est sélectionné dans **Initials**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
Supposons que vous avez sélectionné tous les noms à l'exception de Sue dans **First name** et que vous n'avez rien sélectionné dans **Initials**. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Données utilisées dans l'exemple :

```
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 - fonction de graphique

GetExcludedCount() permet de déterminer le nombre de valeurs exclues dans le champ identifié. Les valeurs exclues incluent les champs alternatifs (en gris clair), exclus (en gris foncé) et sélectionnés exclus (gris foncé avec une coche).

Syntaxe :

```
GetExcludedCount (field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Arguments	Description
field_name	Champ contenant la plage de données à mesurer.

Exemples et résultats :

L'exemple suivant utilise trois champs chargés dans des volets de filtre différents, un pour le prénom **First name**, un autre pour le nom de famille **Last name** et un autre encore pour les initiales **Initials**.

Exemples et résultats

Exemples	Résultats
Si aucune valeur n'est sélectionnée sous **First name**.	`GetExcludedCount (Initials) = 0` Il n'y a pas de sélections.
Si **John** est sélectionné sous **First name**.	`GetExcludedCount (Initials) = 5` On observe 5 valeurs exclues en gris foncé sous **Initials**. La sixième cellule (JA) est en blanc, car elle est associée à la sélection John sous **First name**.
Si **John** et **Peter** sont sélectionnés.	`GetExcludedCount (Initials) = 3` John est associé à 1 valeur, tandis que Peter est associé à 2 valeurs, sous **Initials**.
Si **John** et **Peter** sont sélectionnés sous **First name**, puis **Franc** sous **Last name**.	`GetExcludedCount ([First name]) = 4` Il y a 4 valeurs exclues sous **First name**, signalées en gris foncé. **GetExcludedCount()** évalue les champs ayant des valeurs exclues, y compris les champs exclus sélectionnés et alternatifs.
Si **John** et **Peter** sont sélectionnés sous **First name**, puis **Franc** et **Anderson** sous **Last name**.	`GetExcludedCount (Initials) = 4` On observe 4 valeurs exclues en gris foncé sous **Initials**. Les deux autres cellules (JA et PF) sont affichées en blanc, car elles sont associées aux sélections John et Peter sous **First name**.
Si **John** et **Peter** sont sélectionnés sous **First name**, puis **Franc** et **Anderson** sous **Last name**.	`GetExcludedCount ([Last name]) = 4` On observe 4 valeurs exclues sous **Initials**. Devonshire est affiché en gris clair, tandis que Brown, Carr et Elliot sont signalés en gris foncé.

Données utilisées dans l'exemple :

```
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 - fonction de graphique

La fonction **GetFieldSelections()** renvoie une chaîne (**string**) avec les sélections actives dans un champ.

Si toutes les valeurs sauf deux, ou sauf une, sont sélectionnées, le format 'NOT x,y' ou 'NOT y' sera utilisé. Si vous sélectionnez toutes les valeurs et que le nombre total de valeurs est supérieur à l'argument max_ values, le texte ALL est renvoyé.

Syntaxe :

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Type de données renvoyé : chaîne

Formats de chaîne renvoyés

Format	Description
'a, b, c'	Si le nombre de valeurs sélectionnées est max_values ou moins, la chaîne renvoyée est une liste des valeurs sélectionnées. Les valeurs sont séparées par value_sep comme délimiteur.
'NOT a, b, c'	Si le nombre de valeurs non sélectionnées est max_values ou moins, la chaîne renvoyée est une liste des valeurs non sélectionnées avec NOT comme préfixe. Les valeurs sont séparées par value_sep comme délimiteur.
'x of y'	x = nombre de valeurs sélectionnées y = nombre total de valeurs Cela est renvoyé lorsque max_values < x < (y - max_values).
'ALL'	Renvoyé si toutes les valeurs sont sélectionnées.
'-'	Renvoyé si aucune valeur n'est sélectionnée.
<search string>	Si vous avez effectué votre sélection via la recherche, la chaîne de recherche est renvoyée.

Arguments :

Arguments

Arguments	Description
field_name	Champ contenant la plage de données à mesurer.
value_sep	Séparateur à placer entre les valeurs de champ. La valeur par défaut est la virgule (,).
max_values	Nombre maximum de valeurs de champ pouvant être listées individuellement. Lorsqu'un plus grand nombre de valeurs est sélectionné, le format « x valeurs sur y » le remplace. La valeur par défaut est 6.
state_name	Nom d'un état alternatif ayant été choisi pour cette visualisation en particulier. Si l'argument **state_name** est utilisé, seules les sélections associées au nom d'état spécifié sont prises en compte.

Exemples et résultats :

Dans l'exemple suivant, le champ **First name** est chargé dans un volet de filtre.

Exemples et résultats

Exemples	Résultats
Supposons que **John** est sélectionné dans **First name**. `GetFieldSelections ([First name])`	'John'
Supposons que **John** et **Peter** sont sélectionnés. `GetFieldSelections ([First name])`	'John,Peter'
Supposons que **John** et **Peter** sont sélectionnés. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Supposons que **John**, **Sue** et **Mark** sont sélectionnés dans **First name**. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter', car la valeur 2 est spécifiée comme la valeur de l'argument max_values. Sinon, le résultat aurait été John; Sue; Mark.

Données utilisées dans l'exemple :

```
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 - fonction de graphique

Cette fonction de graphique renvoie le nombre de valeurs non sélectionnées dans le champ intitulé **fieldname**. Le champ doit être en mode And pour que cette fonction soit pertinente.

Syntaxe :

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Arguments :

Arguments

Argument	Description
fieldname	Nom du champ à évaluer.
includeexcluded	Si l'argument **includeexcluded** est défini sur True, le nombre inclut les valeurs sélectionnées exclues par les sélections effectuées dans un autre champ.

Exemples :

```
GetNotSelectedCount( Country )
GetNotSelectedCount( Country, true )
```

GetObjectDimension - fonction de graphique

GetObjectDimension() renvoie le nom de la dimension. **Index** désigne un entier facultatif qui indique la dimension à renvoyer.

Il n'est pas possible d'utiliser cette fonction dans un graphique aux emplacements suivants : titre, sous-titre, pied de page et expression de ligne de référence.

Il n'est pas possible de faire référence au nom d'une dimension ou d'une mesure dans un autre objet à l'aide du paramètre Object ID.

Syntaxe :

```
GetObjectDimension ([index])
```

Exemple :

```
GetObjectDimension(1)
```

Exemple : Expression de graphique

Table Qlik Sense affichant des exemples de la fonction GetObjectDimension utilisée dans une expression de graphique

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Si vous souhaitez renvoyer le nom d'une mesure, utilisez plutôt la fonction **GetObjectMeasure**.

GetObjectField - fonction de graphique

GetObjectField() renvoie le nom de la dimension. **Index** désigne un entier facultatif qui indique la dimension à renvoyer.

Il n'est pas possible d'utiliser cette fonction dans un graphique aux emplacements suivants : titre, sous-titre, pied de page et expression de ligne de référence.

Il n'est pas possible de faire référence au nom d'une dimension ou d'une mesure dans un autre objet à l'aide du paramètre Object ID.

Syntaxe :

```
GetObjectField ([index])
```

Exemple :

```
GetObjectField(1)
```

Exemple : Expression de graphique

Table Qlik Sense affichant des exemples de la fonction GetObjectField utilisée dans une expression de graphique.

transaction_date	customer_id	transaction_quantity	=GetObjectField ()	=GetObjectField (0)	=GetObjectField (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Si vous souhaitez renvoyer le nom d'une mesure, utilisez plutôt la fonction **GetObjectMeasure**.

GetObjectMeasure - fonction de graphique

GetObjectMeasure() renvoie le nom de la mesure. **Index** désigne un entier facultatif qui indique la mesure à renvoyer.

Il n'est pas possible d'utiliser cette fonction dans un graphique aux emplacements suivants : titre, sous-titre, pied de page et expression de ligne de référence.

Il n'est pas possible de faire référence au nom d'une dimension ou d'une mesure dans un autre objet à l'aide du paramètre Object ID.

Syntaxe :

```
GetObjectMeasure ([index])
```

Exemple :

```
GetObjectMeasure(1)
```

Exemple : Expression de graphique

Table Qlik Sense affichant des exemples de la fonction GetObjectMeasure utilisée dans une expression de graphique

customer_id	sum (transaction_quantity)	Avg (transaction_quantity)	=GetObjectMeasure ()	=GetObjectMeasure(0)	=GetObjectMeasure(1)
49681	13	13	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)
203521	27	13.5	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)

Si vous souhaitez renvoyer le nom d'une dimension, utilisez plutôt la fonction **GetObjectField**.

GetPossibleCount - fonction de graphique

GetPossibleCount() permet de déterminer le nombre de valeurs possibles dans le champ identifié. Si le champ identifié inclut des sélections, les champs sélectionnés (en vert) sont pris en compte. Sinon, ce sont les valeurs associées (en blanc) qui sont comptabilisées. .

Si les champs comportent des sélections, **GetPossibleCount()** renvoie le nombre de champs sélectionnés (en vert).

Type de données renvoyé : entier

Syntaxe :

```
GetPossibleCount (field_name)
```

Arguments :

Arguments

Arguments	Description
field_name	Champ contenant la plage de données à mesurer.

Exemples et résultats :

L'exemple suivant utilise deux champs chargés dans des volets de filtre différents, un pour le prénom **First name** et l'autre pour les initiales **Initials**.

Exemples et résultats

Exemples	Résultats
Supposons que **John** est sélectionné dans **First name**. `GetPossibleCount ([Initials])`	1, car la liste de sélection Initials comporte 1 valeur associée à la sélection, **John**, dans **First name**.
Supposons que **John** est sélectionné dans **First name**. `GetPossibleCount ([First name])`	1, car il y a 1 sélection, **John**, dans **First name**.
Supposons que **Peter** est sélectionné dans **First name**. `GetPossibleCount ([Initials])`	2, car Peter est associé à 2 valeurs dans **Initials**.
Supposons qu'aucune valeur n'est sélectionnée dans **First name**. `GetPossibleCount ([First name])`	5, car aucune sélection n'a été effectuée et qu'il y a 5 valeurs uniques dans **First name**.
Supposons qu'aucune valeur n'est sélectionnée dans **First name**. `GetPossibleCount ([Initials])`	6, car aucune sélection n'a été effectuée et qu'il y a 6 valeurs uniques dans **Initials**.

Données utilisées dans l'exemple :

```
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 - fonction de graphique

GetSelectedCount() permet de déterminer le nombre de valeurs sélectionnées (en vert) d'un champ.

Syntaxe :

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Type de données renvoyé : entier

Arguments :

Arguments

Arguments	Description
field_name	Champ contenant la plage de données à mesurer.
include_ excluded	Si cet argument est défini sur **True()**, le décompte inclut les valeurs sélectionnées, qui sont actuellement exclues par des sélections dans d'autres champs. Si l'argument est défini sur False ou s'il est omis, ces valeurs sont exclues.
state_name	Nom d'un état alternatif ayant été choisi pour cette visualisation en particulier. Si l'argument **state_name** est utilisé, seules les sélections associées au nom d'état spécifié sont prises en compte.

Exemples et résultats :

L'exemple suivant utilise trois champs chargés dans des volets de filtre différents, un pour le prénom **First name**, un autre pour les initiales **Initials** et un autre encore pour les propriétaires de téléphone portable **Has cellphone**.

Exemples et résultats

Exemples	Résultats
Supposons que **John** est sélectionné dans **First name**. `GetSelectedCount ([First name])`	1, car une valeur est sélectionnée dans **First name**.
Supposons que **John** est sélectionné dans **First name**. `GetSelectedCount ([Initials])`	0, car aucune valeur n'est sélectionnée dans **Initials**.
Si aucune sélection n'est effectuée dans **First name**, sélectionnez toutes les valeurs contenues dans **Initials**, puis choisissez la valeur **Yes** dans **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Même si les sélections comportant les initiales MC et PD sous **Initials** ont le champ **Has cellphone** défini sur **No**, le résultat demeure 6, car l'argument include_ excluded est défini sur True().

Données utilisées dans l'exemple :

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

5.10 Fonctions de fichier

Les fonctions de fichier (disponibles uniquement dans les expressions de script) renvoient des informations sur le fichier de table en cours de lecture. Elles renvoient la valeur NULL pour toutes les sources de données à l'exception des fichiers de table (exception : **ConnectString( )**).

Vue d'ensemble des fonctions de fichier

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Attribute

Cette fonction de script renvoie sous forme de texte la valeur des balises méta de différents fichiers multimédia. Les formats de fichier suivants sont pris en charge : MP3, WMA, WMV, PNG et JPG. Si le fichier **filename** n'existe pas, si son format n'est pas pris en charge ou s'il ne contient pas de balise méta nommée **attributename**, la valeur NULL est renvoyée.

```
Attribute (filename, attributename)
```

ConnectString

La fonction **ConnectString()** renvoie le nom de la connexion de données active pour les connexions ODBC ou OLE DB. La fonction renvoie une chaîne vide si aucune instruction **connect** n'a été exécutée ou après une instruction **disconnect**.

```
ConnectString ()
```

FileBaseName

La fonction **FileBaseName** renvoie une chaîne contenant le nom du fichier de table en cours de lecture, sans chemin d'accès ni extension.

```
FileBaseName ()
```

FileDir

La fonction **FileDir** renvoie une chaîne contenant le chemin d'accès au répertoire dans lequel figure le fichier de table en cours de lecture.

```
FileDir ()
```

FileExtension

La fonction **FileExtension** renvoie une chaîne contenant l'extension du fichier de table en cours de lecture.

```
FileExtension ()
```

FileName

La fonction **FileName** renvoie une chaîne contenant le nom du fichier de table en cours de lecture, sans chemin d'accès mais avec l'extension.

```
FileName ()
```

FilePath

La fonction **FilePath** renvoie une chaîne contenant le chemin d'accès complet au fichier de table en cours de lecture.

```
FilePath ()
```

FileSize

La fonction **FileSize** renvoie un entier contenant la taille en octets du fichier filename ou, si aucun argument filename n'est spécifié, du fichier de table en cours de lecture.

```
FileSize ()
```

FileTime

La fonction **FileTime** renvoie un horodatage au format UTC pour la date et l'heure de la dernière modification apportée au fichier filename. Si aucun argument filename n'est spécifié, la fonction se réfère au fichier de table en cours de lecture.

```
FileTime ([ filename ])
```

GetFolderPath

La fonction **GetFolderPath** renvoie la valeur de la fonction Microsoft Windows *SHGetFolderPath*. Cette fonction utilise comme données d'entrée le nom d'un dossier Microsoft Windows et renvoie le chemin d'accès complet au dossier.

```
GetFolderPath ()
```

QvdCreateTime

Cette fonction de script renvoie l'horodatage de l'en-tête XML à partir d'un fichier QVD, le cas échéant. Dans le cas contraire, la valeur NULL est renvoyée. Dans l'horodatage, l'heure est indiquée dans le fuseau horaire UTC.

```
QvdCreateTime (filename)
```

QvdFieldName

Cette fonction de script renvoie le nom du numéro de champ **fieldno** contenu dans un fichierQVD. S'il n'existe pas de champ, NULL est renvoyé.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Cette fonction de script renvoie le nombre de champs contenus dans un fichier QVD.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Cette fonction de script renvoie le nombre d'enregistrements contenus dans un fichier QVD.

```
QvdNoOfRecords (filename)
```

QvdTableName

Cette fonction de script renvoie le nom de la table stockée dans un fichier QVD.

```
QvdTableName (filename)
```

Attribute

Cette fonction de script renvoie sous forme de texte la valeur des balises méta de différents fichiers multimédia. Les formats de fichier suivants sont pris en charge : MP3, WMA, WMV, PNG et JPG. Si le fichier **filename** n'existe pas, si son format n'est pas pris en charge ou s'il ne contient pas de balise méta nommée **attributename**, la valeur NULL est renvoyée.

Syntaxe :

```
Attribute(filename, attributename)
```

Il est possible de lire un grand nombre de balises méta. Les exemples de cette rubrique montrent les balises qui peuvent être lues par les différents types de fichier pris en charge.

*Vous pouvez uniquement lire les balises méta enregistrées dans le fichier conformément à la spécification pertinente, par exemple ID2v3 pour les fichiers MP3 ou EXIF pour les fichiers JPG, et non les informations méta enregistrées dans l'**Explorateur de fichiers Windows**.*

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier multimédia incluant, le cas échéant, son chemin d'accès, tel qu'une connexion de données de type dossier. **Exemple :** *'lib://Table Files/'* En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data*
attributename	Nom d'une balise méta.

Dans ces exemples, la fonction **GetFolderPath** est utilisée pour rechercher les chemins d'accès aux fichiers multimédia. Comme la fonction **GetFolderPath** est uniquement prise en charge en mode hérité, vous devez remplacer les références à **GetFolderPath** par un chemin de connexion de données lib:// lorsque vous utilisez cette fonction en mode standard ou dans Qlik Sense SaaS.

Restrictions d'accès au système de fichiers (page 1473)

Example 1: Fichiers MP3

Ce script permet de lire toutes les balises méta MP3 possibles contenues dans le dossier *MyMusic*.

```
// Script to read MP3 meta tags for each vExt in 'mp3' for each vFoundFile in filelist(
GetFolderPath('MyMusic') & '\*.'& vExt ) FileList: LOAD FileLongName,     subfield
(FileLongName,'\',-1) as FileShortName,     num(FileSize(FileLongName),'# ### ### ###',',','
') as FileSize,     FileTime(FileLongName) as FileTime,      // ID3v1.0 and ID3v1.1 tags
Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName, 'Artist') as Artist,
 Attribute(FileLongName, 'Album') as Album,     Attribute(FileLongName, 'Year') as Year,
Attribute(FileLongName, 'Comment') as Comment,     Attribute(FileLongName, 'Track') as Track,
   Attribute(FileLongName, 'Genre') as Genre,

     // ID3v2.3 tags     Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
Attribute(FileLongName, 'APIC') as APIC, // Attached picture     Attribute(FileLongName,
'COMM') as COMM, // Comments     Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
   Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration     Attribute
(FileLongName, 'EQUA') as EQUA, // Equalization     Attribute(FileLongName, 'ETCO') as ETCO,
// Event timing codes     Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated
object     Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list     Attribute(FileLongName,
'LINK') as LINK, // Linked information     Attribute(FileLongName, 'MCDI') as MCDI, // Music
```

```
CD identifier     Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame     Attribute(FileLongName,
'PRIV') as PRIV, // Private frame     Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
    Attribute(FileLongName, 'POPM') as POPM, // Popularimeter

     Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame     Attribute
(FileLongName, 'RBUF') as RBUF, // Recommended buffer size     Attribute(FileLongName, 'RVAD')
as RVAD, // Relative volume adjustment     Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
   Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text     Attribute
(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes     Attribute(FileLongName,
'TALB') as TALB, // Album/Movie/Show title     Attribute(FileLongName, 'TBPM') as TBPM, // BPM
(beats per minute)     Attribute(FileLongName, 'TCOM') as TCOM, // Composer     Attribute
(FileLongName, 'TCON') as TCON, // Content type     Attribute(FileLongName, 'TCOP') as TCOP,
// Copyright message     Attribute(FileLongName, 'TDAT') as TDAT, // Date     Attribute
(FileLongName, 'TDLY') as TDLY, // Playlist delay

     Attribute(FileLongName, 'TENC') as TENC, // Encoded by     Attribute(FileLongName,
'TEXT') as TEXT, // Lyricist/Text writer     Attribute(FileLongName, 'TFLT') as TFLT, // File
type     Attribute(FileLongName, 'TIME') as TIME, // Time     Attribute(FileLongName, 'TIT1')
as TIT1, // Content group description     Attribute(FileLongName, 'TIT2') as TIT2, //
Title/songname/content description     Attribute(FileLongName, 'TIT3') as TIT3, //
Subtitle/Description refinement     Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
 Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)     Attribute(FileLongName, 'TLEN')
as TLEN, // Length     Attribute(FileLongName, 'TMED') as TMED, // Media type

     Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title     Attribute
(FileLongName, 'TOFN') as TOFN, // Original filename     Attribute(FileLongName, 'TOLY') as
TOLY, // Original lyricist(s)/text writer(s)     Attribute(FileLongName, 'TOPE') as TOPE, //
Original artist(s)/performer(s)     Attribute(FileLongName, 'TORY') as TORY, // Original
release year     Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee     Attribute
(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)     Attribute(FileLongName,
'TPE2') as TPE2, // Band/orchestra/accompaniment

     Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement     Attribute
(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set     Attribute(FileLongName, 'TPUB')
as TPUB, // Publisher     Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in
set     Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates     Attribute
(FileLongName, 'TRSN') as TRSN, // Internet radio station name     Attribute(FileLongName,
'TRSO') as TRSO, // Internet radio station owner

     Attribute(FileLongName, 'TSIZ') as TSIZ, // Size     Attribute(FileLongName, 'TSRC') as
TSRC, // ISRC (international standard recording code)     Attribute(FileLongName, 'TSSE') as
TSSE, // Software/Hardware and settings used for encoding     Attribute(FileLongName, 'TYER')
as TYER, // Year     Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information
frame     Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier     Attribute
(FileLongName, 'USER') as USER, // Terms of use     Attribute(FileLongName, 'USLT') as USLT,
// Unsychronized lyric/text transcription     Attribute(FileLongName, 'WCOM') as WCOM, //
Commercial information     Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal
information

     Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage     Attribute
(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage     Attribute
(FileLongName, 'WOAS') as WOAS, // Official audio source webpage     Attribute(FileLongName,
'WORS') as WORS, // Official internet radio station homepage     Attribute(FileLongName,
```

```
'WPAY') as WPAY, // Payment     Attribute(FileLongName, 'WPUB') as WPUB, // Publishers
official webpage     Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 2: JPEG

Ce script permet de lire toutes les balises méta EXIF possibles contenues dans les fichiers JPG du dossier *MyPictures*.

```
// Script to read Jpeg Exif meta tags for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif',
'jfi' for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      // ************   Exif Main (IFD0) Attributes   ************     Attribute
(FileLongName, 'ImageWidth') as ImageWidth,     Attribute(FileLongName, 'ImageLength') as
ImageLength,     Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,     Attribute
(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,      Attribute
(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
Attribute(FileLongName, 'ImageDescription') as ImageDescription,     Attribute(FileLongName,
'Make') as Make,     Attribute(FileLongName, 'Model') as Model,     Attribute(FileLongName,
'StripOffsets') as StripOffsets,     Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,      Attribute(FileLongName,
'SamplesPerPixel') as SamplesPerPixel,     Attribute(FileLongName, 'RowsPerStrip') as
RowsPerStrip,     Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,     Attribute
(FileLongName, 'XResolution') as XResolution,     Attribute(FileLongName, 'YResolution') as
YResolution,     Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,      Attribute(FileLongName,
'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,      Attribute(FileLongName,
'TransferFunction') as TransferFunction,     Attribute(FileLongName, 'Software') as Software,
   Attribute(FileLongName, 'DateTime') as DateTime,     Attribute(FileLongName, 'Artist') as
Artist,     Attribute(FileLongName, 'HostComputer') as HostComputer,     Attribute
(FileLongName, 'WhitePoint') as WhitePoint,     Attribute(FileLongName,
'PrimaryChromaticities') as PrimaryChromaticities,     Attribute(FileLongName,
'YCbCrCoefficients') as YCbCrCoefficients,     Attribute(FileLongName, 'YCbCrSubSampling') as
YCbCrSubSampling,     Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,      Attribute(FileLongName, 'ReferenceBlackWhite')
as ReferenceBlackWhite,     Attribute(FileLongName, 'Rating') as Rating,     Attribute
(FileLongName, 'RatingPercent') as RatingPercent,     Attribute(FileLongName,
'ThumbnailFormat') as ThumbnailFormat,
```

```
    //  examples: 0=Raw Rgb, 1=Jpeg,      Attribute(FileLongName, 'Copyright') as Copyright,
  Attribute(FileLongName, 'ExposureTime') as ExposureTime,     Attribute(FileLongName,
'FNumber') as FNumber,     Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,     Attribute(FileLongName,
'TimeZoneOffset') as TimeZoneOffset,     Attribute(FileLongName, 'SensitivityType') as
SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
     Attribute(FileLongName, 'ExifVersion') as ExifVersion,     Attribute(FileLongName,
'DateTimeOriginal') as DateTimeOriginal,     Attribute(FileLongName, 'DateTimeDigitized') as
DateTimeDigitized,     Attribute(FileLongName, 'ComponentsConfiguration') as
ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,      Attribute(FileLongName,
'CompressedBitsPerPixel') as CompressedBitsPerPixel,     Attribute(FileLongName,
'ShutterSpeedValue') as ShutterSpeedValue,     Attribute(FileLongName, 'ApertureValue') as
ApertureValue,     Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //
examples: -1=Unknown,      Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
   Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,     Attribute
(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,      Attribute(FileLongName, 'MeteringMode') as
MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,      Attribute(FileLongName,
'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
Attribute(FileLongName, 'Flash') as Flash,     Attribute(FileLongName, 'FocalLength') as
FocalLength,     Attribute(FileLongName, 'SubjectArea') as SubjectArea,     Attribute
(FileLongName, 'MakerNote') as MakerNote,     Attribute(FileLongName, 'UserComment') as
UserComment,     Attribute(FileLongName, 'SubSecTime') as SubSecTime,
```

```
    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,    Attribute
(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,    Attribute(FileLongName,
'XPTitle') as XPTitle,    Attribute(FileLongName, 'XPComment') as XPComment,

    Attribute(FileLongName, 'XPAuthor') as XPAuthor,    Attribute(FileLongName,
'XPKeywords') as XPKeywords,    Attribute(FileLongName, 'XPSubject') as XPSubject,
Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,    Attribute(FileLongName,
'ColorSpace') as ColorSpace, //  examples: 1=sRGB, 65535=Uncalibrated,    Attribute
(FileLongName, 'PixelXDimension') as PixelXDimension,    Attribute(FileLongName,
'PixelYDimension') as PixelYDimension,    Attribute(FileLongName, 'RelatedSoundFile') as
RelatedSoundFile,

    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,    Attribute
(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,    Attribute(FileLongName,
'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

   //  examples: 1=None, 2=Inch, 3=Centimeter,    Attribute(FileLongName, 'ExposureIndex')
as ExposureIndex,    Attribute(FileLongName, 'SensingMethod') as SensingMethod,

   //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

   // 4=Three-chip color area sensor, 5=Color sequential area sensor,

   // 7=Trilinear sensor, 8=Color sequential linear sensor,    Attribute(FileLongName,
'FileSource') as FileSource,

   //  examples: 0=Other, 1=Scanner of transparent type,

   // 2=Scanner of reflex type, 3=Digital still camera,    Attribute(FileLongName,
'SceneType') as SceneType,

   //  examples: 1=A directly photographed image,    Attribute(FileLongName, 'CFAPattern')
as CFAPattern,    Attribute(FileLongName, 'CustomRendered') as CustomRendered,

   //  examples: 0=Normal process, 1=Custom process,    Attribute(FileLongName,
'ExposureMode') as ExposureMode,

   //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,    Attribute
(FileLongName, 'WhiteBalance') as WhiteBalance,

   //  examples: 0=Auto white balance, 1=Manual white balance,    Attribute(FileLongName,
'DigitalZoomRatio') as DigitalZoomRatio,    Attribute(FileLongName, 'FocalLengthIn35mmFilm')
as FocalLengthIn35mmFilm,    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

   //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,    Attribute
(FileLongName, 'GainControl') as GainControl,

   //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
  Attribute(FileLongName, 'Contrast') as Contrast,

   //  examples: 0=Normal, 1=Soft, 2=Hard,    Attribute(FileLongName, 'Saturation') as
Saturation,

   //  examples: 0=Normal, 1=Low saturation, 2=High saturation,    Attribute(FileLongName,
'Sharpness') as Sharpness,
```

```
    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName,
'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,      Attribute
(FileLongName, 'ImageUniqueID') as ImageUniqueID,     Attribute(FileLongName,
'BodySerialNumber') as BodySerialNumber,     Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_
GAMMA,     Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,     Attribute
(FileLongName, 'OffsetSchema') as OffsetSchema,

     // ************   Interoperability Attributes   ************     Attribute(FileLongName,
'InteroperabilityIndex') as InteroperabilityIndex,     Attribute(FileLongName,
'InteroperabilityVersion') as InteroperabilityVersion,     Attribute(FileLongName,
'InteroperabilityRelatedImageFileFormat') as InteroperabilityRelatedImageFileFormat,
Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,     Attribute(FileLongName,
'InteroperabilityRelatedImageLength') as InteroperabilityRelatedImageLength,     Attribute
(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute(FileLongName,
'InteroperabilityPrintImageMatching') as InteroperabilityPrintImageMatching,     //
************   GPS Attributes   ************     Attribute(FileLongName, 'GPSVersionID') as
GPSVersionID,     Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,     Attribute
(FileLongName, 'GPSLatitude') as GPSLatitude,     Attribute(FileLongName, 'GPSLongitudeRef')
as GPSLongitudeRef,     Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,     Attribute
(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,      Attribute(FileLongName,
'GPSAltitude') as GPSAltitude,     Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
 Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,     Attribute(FileLongName,
'GPSStatus') as GPSStatus,     Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
 Attribute(FileLongName, 'GPSDOP') as GPSDOP,     Attribute(FileLongName, 'GPSSpeedRef') as
GPSSpeedRef,

     Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,     Attribute(FileLongName,
'GPSTrackRef') as GPSTrackRef,     Attribute(FileLongName, 'GPSTrack') as GPSTrack,
Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,     Attribute
(FileLongName, 'GPSImgDirection') as GPSImgDirection,     Attribute(FileLongName,
'GPSMapDatum') as GPSMapDatum,     Attribute(FileLongName, 'GPSDestLatitudeRef') as
GPSDestLatitudeRef,

     Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,     Attribute
(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,     Attribute(FileLongName,
'GPSDestLongitude') as GPSDestLongitude,     Attribute(FileLongName, 'GPSDestBearingRef') as
GPSDestBearingRef,     Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

     Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,     Attribute
(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,     Attribute(FileLongName,
'GPSAreaInformation') as GPSAreaInformation,     Attribute(FileLongName, 'GPSDateStamp') as
GPSDateStamp,     Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,  LOAD @1:n as FileLongName
Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 3: Fichiers multimédia Windows

Ce script permet de lire toutes les balises méta WMA/WMV ASF possibles contenues dans le dossier *MyMusic*.

```
// Script to read WMA/WMV ASF meta tags for each vExt in 'asf', 'wma', 'wmv' for each
vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName,
'Author') as Author,     Attribute(FileLongName, 'Copyright') as Copyright,     Attribute
(FileLongName, 'Description') as Description,

     Attribute(FileLongName, 'Rating') as Rating,     Attribute(FileLongName, 'PlayDuration')
as PlayDuration,     Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,     Attribute(FileLongName,
'WMFSDKNeeded') as WMFSDKNeeded,     Attribute(FileLongName, 'IsVBR') as IsVBR,     Attribute
(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,

     Attribute(FileLongName, 'PeakValue') as PeakValue,     Attribute(FileLongName,
'AverageLevel') as AverageLevel; LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no
labels); Next vFoundFile Next vExt
```

Example 4: PNG

Ce script permet de lire toutes les balises méta PNG possibles contenues dans le dossier *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

La fonction **ConnectString()** renvoie le nom de la connexion de données active pour les connexions ODBC ou OLE DB. La fonction renvoie une chaîne vide si aucune instruction **connect** n'a été exécutée ou après une instruction **disconnect**.

Syntaxe :

```
ConnectString()
```

Exemples et résultats :

Exemples de script

Exemple	Résultat

`LIB CONNECT TO 'Tutorial ODBC';`
`ConnectString:`
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Renvoie 'Tutorial ODBC' dans le champ ConnectString.
	
	Dans cet exemple, nous partons du principe que vous disposez d'une connexion de données intitulée Tutorial ODBC.

FileBaseName

La fonction **FileBaseName** renvoie une chaîne contenant le nom du fichier de table en cours de lecture, sans chemin d'accès ni extension.

Syntaxe :

```
FileBaseName()
```

Exemples et résultats :

Exemples de script

Exemple	Résultat
`LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt`	Renvoie abc dans le champ X de chaque enregistrement lu.

FileDir

La fonction **FileDir** renvoie une chaîne contenant le chemin d'accès au répertoire dans lequel figure le fichier de table en cours de lecture.

Syntaxe :

```
FileDir()
```

Cette fonction prend uniquement en charge les connexions de données de type dossier en mode standard.

Exemples et résultats :

Exemples de script

Exemple	Résultat

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Renvoie C:\UserFiles dans le champ X de chaque enregistrement lu.

FileExtension

La fonction **FileExtension** renvoie une chaîne contenant l'extension du fichier de table en cours de lecture.

Syntaxe :

```
FileExtension()
```

Exemples et résultats :

Exemples de script

Exemple	Résultat
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Renvoie txt dans le champ X de chaque enregistrement lu.

FileName

La fonction **FileName** renvoie une chaîne contenant le nom du fichier de table en cours de lecture, sans chemin d'accès mais avec l'extension.

Syntaxe :

```
FileName()
```

Exemples et résultats :

Exemples de script

Exemple	Résultat
`LOAD *, FileName( ) as X from C:\UserFiles\abc.txt`	Renvoie 'abc.txt' dans le champ X de chaque enregistrement lu.

FilePath

La fonction **FilePath** renvoie une chaîne contenant le chemin d'accès complet au fichier de table en cours de lecture.

Syntaxe :

```
FilePath()
```

Cette fonction prend uniquement en charge les connexions de données de type dossier en mode standard.

Exemples et résultats :

Exemples de script

Exemple	Résultat

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Renvoie 'C:\UserFiles\abc.txt' dans le champ X de chaque enregistrement lu.

FileSize

La fonction **FileSize** renvoie un entier contenant la taille en octets du fichier filename ou, si aucun argument filename n'est spécifié, du fichier de table en cours de lecture.

Syntaxe :

FileSize([filename]**)**

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. Si vous ne précisez pas de nom de fichier, le fichier de table en cours de lecture est utilisé. **Exemple :** ***'lib://Table Files/'*** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** ***c:\data*** • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** ***data*** • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** ***http://www.qlik.com***

Exemples et résultats :

Exemples de script

Exemple	Résultat
`LOAD *, FileSize( ) as X from abc.txt;`	Renvoie la taille du fichier spécifié (abc.txt) sous la forme d'un entier dans le champ X de chaque enregistrement lu.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Renvoie la taille du fichier xyz.xls.

FileTime

La fonction **FileTime** renvoie un horodatage au format UTC pour la date et l'heure de la dernière modification apportée au fichier filename. Si aucun argument filename n'est spécifié, la fonction se réfère au fichier de table en cours de lecture.

Syntaxe :

```
FileTime([ filename ])
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** ***'lib://Table Files/'*** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** ***c:\data*** • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** ***data*** • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** ***http://www.qlik.com***

Exemples et résultats :

Exemples de script

Exemple	Résultat
`LOAD *, FileTime( ) as X from abc.txt;`	Renvoie la date et l'heure de la dernière modification du fichier (abc.txt) sous forme d'horodatage dans le champ X de chaque enregistrement lu.
FileTime('xyz.xls')	Renvoie l'horodatage de la dernière modification du fichier xyz.xls.

GetFolderPath

La fonction **GetFolderPath** renvoie la valeur de la fonction Microsoft Windows *SHGetFolderPath*. Cette fonction utilise comme données d'entrée le nom d'un dossier Microsoft Windows et renvoie le chemin d'accès complet au dossier.

Cette fonction n'est pas prise en charge en mode standard.

Syntaxe :

```
GetFolderPath(foldername)
```

Arguments :

Arguments

Argument	Description
foldername	Nom du dossier Microsoft Windows. Le nom du dossier ne doit contenir aucun espace. Il convient de supprimer tous les espaces présents dans les noms de dossier affichés dans Windows Explorer. Exemples : *MyMusic* *MyDocuments*

Exemples et résultats :

Cet exemple a pour objectif d'obtenir les chemins d'accès aux dossiers Microsoft Windows suivants : *MyMusic*, *MyPictures* et *Windows*. Ajoutez l'exemple de script à votre application et rechargez cette dernière.

```
LOAD   GetFolderPath('MyMusic') as MyMusic,  GetFolderPath('MyPictures') as MyPictures,
GetFolderPath('Windows') as Windows AutoGenerate 1;
```

Une fois l'application rechargée, les champs *MyMusic*, *MyPictures* et *Windows* sont ajoutés au modèle de données. Chaque champ contient le chemin d'accès au dossier défini dans les données d'entrée. Par exemple :

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Cette fonction de script renvoie l'horodatage de l'en-tête XML à partir d'un fichier QVD, le cas échéant. Dans le cas contraire, la valeur NULL est renvoyée. Dans l'horodatage, l'heure est indiquée dans le fuseau horaire UTC.

Syntaxe :

```
QvdCreateTime(filename)
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier QVD, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** ***'lib://Table Files/'*** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** ***c:\data*** • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** ***data*** • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** ***http://www.qlik.com***

Exemple :

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Cette fonction de script renvoie le nom du numéro de champ **fieldno** contenu dans un fichierQVD. S'il n'existe pas de champ, NULL est renvoyé.

Syntaxe :

```
QvdFieldName(filename , fieldno)
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier QVD, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** *'lib://Table Files/'* En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data* • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** *http://www.qlik.com*
fieldno	Numéro du champ inclus dans la table que contient le fichier QVD.

Exemples :

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Les trois exemples renvoient tous le nom du cinquième champ de la table contenue dans le fichier QVD.

QvdNoOfFields

Cette fonction de script renvoie le nombre de champs contenus dans un fichier QVD.

Syntaxe :

```
QvdNoOfFields(filename)
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier QVD, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** *'lib://Table Files/'* En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data* • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** *http://www.qlik.com*

Exemples :

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Exemple : Cette fonction de script renvoie le nombre d'enregistrements contenus dans un fichier QVD.

Syntaxe :

```
QvdNoOfRecords(filename)
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier QVD, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** *'lib://Table Files/'* En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** *c:\data* • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** *data* • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** *http://www.qlik.com*

Exemples :

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Cette fonction de script renvoie le nom de la table stockée dans un fichier QVD.

Syntaxe :

```
QvdTableName(filename)
```

Arguments :

Arguments

Argument	Description
filename	Nom d'un fichier QVD, comprenant éventuellement le chemin d'accès, sous forme de connexion de données de type dossier ou fichier Web. **Exemple :** ***'lib://Table Files/'*** En langage de script, les formats de chemin d'accès suivants sont également pris en charge en mode hérité : • absolu **Exemple :** ***c:\data*** • chemin d'accès relatif au répertoire de travail de l'application Qlik Sense **Exemple :** ***data*** • adresse URL (HTTP ou FTP), renvoyant à un emplacement sur Internet ou un intranet **Exemple :** ***http://www.qlik.com***

Exemples :

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Fonctions financières

Les fonctions financières s'utilisent dans le script de chargement de données et dans les expressions de graphique pour calculer des paiements et des taux d'intérêt.
Pour tous les arguments, l'argent versé est représenté par des nombres négatifs. L'argent perçu est représenté par des nombres positifs.
Voici une liste des arguments utilisés dans les fonctions financières (excepté celles qui commencent par **range**-).

> *Pour toutes les fonctions financières, il est extrêmement important de rester cohérent dans les unités utilisées pour les arguments **rate** et **nper**. Si vous effectuez des paiements mensuels pour un prêt contracté sur cinq ans à un taux d'intérêt annuel de 6 %, utilisez 0.005 (6 %/12) pour le taux (**rate**) et 60 (5*12) pour le nombre d'échéances (**nper**). Si vous effectuez des paiements annuels pour le même prêt, utilisez 6 % pour le taux (**rate**) et 5 pour le nombre d'échéances (**nper**).*

Vue d'ensemble des fonctions financières

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

FV

Cette fonction renvoie la valeur future d'un investissement sur la base de paiements périodiques constants et d'un intérêt annuel simple.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Cette fonction renvoie le nombre d'échéances d'un investissement sur la base de paiements périodiques constants et d'un taux d'intérêt constant.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Cette fonction renvoie le paiement d'un emprunt sur la base de paiements périodiques constants et d'un taux d'intérêt constant. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Cette fonction renvoie la valeur présente d'un investissement.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Cette fonction renvoie le taux d'intérêt par période de l'annuité. Le résultat suit le format de nombre par défaut de la fonction **Fix** : deux décimales et en %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Le modèle Black and Scholes est un modèle mathématique conçu pour les produits dérivés des marchés financiers. La formule calcule la valeur théorique d'une option. Dans Qlik Sense, la fonction **BlackAndSchole** renvoie la valeur selon la formule Black and Scholes non modifiée (options de style européen).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate ,
type)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
strike	Futur prix d'achat du titre.
time_left	Nombre de périodes restant.
underlying_ price	Valeur actuelle du titre.
vol	Volatilité (du cours du titre) exprimée sous forme de pourcentage décimal, par période.
risk_free_rate	Taux hors risque exprimé sous forme de pourcentage décimal, par période.
call_or_put	Type d'option : 'c', 'call' ou toute valeur numérique autre que zéro pour les options d'achat call. 'p', 'put' ou 0 pour les options de vente put.

Limitations :

La valeur de strike, de time_left et de underlying_price doit être > 0.

La valeur de vol et de risk_free_rate doit être : < 0 ou > 0.

Exemples et résultats :

Exemples de script

Exemple	Résultat
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` Il s'agit du prix théorique d'une option d'achat d'un titre qui vaut aujourd'hui 68.5 à un prix de 130 dans 4 ans. La formule utilise une volatilité de 0.4 (40 %) par an et un taux d'intérêt hors risque de 0.04 (4 %).	Renvoie 11.245.

FV

Cette fonction renvoie la valeur future d'un investissement sur la base de paiements périodiques constants et d'un intérêt annuel simple.

Syntaxe :

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise..

Arguments :

Arguments

Argument	Description
rate	Taux d'intérêt par période.
nper	Nombre total d'échéances de paiement dans une annuité.
pmt	Paiement effectué à chaque échéance. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.
pv	Valeur, ou montant forfaitaire, que vaut une série de paiements à venir au moment présent. Si **pv** est omis, la fonction utilise 0 (zéro).
type	Doit être égal à 0 si les paiements sont dus à la fin de la période et égal à 1 si les paiements sont dus au début de la période. Si **type** est omis, la fonction utilise 0.

Exemples et résultats :

Exemple de script

Exemple	Résultat
Vous payez un nouvel équipement ménager en 36 mensualités de $20. Le taux d'intérêt annuel est de 6 %. La facture arrive à la fin de chaque mois. Quel est le montant total investi une fois la dernière facture réglée ? `FV(0.005,36,-20)`	Renvoie $786.72.

nPer

Cette fonction renvoie le nombre d'échéances d'un investissement sur la base de paiements périodiques constants et d'un taux d'intérêt constant.

Syntaxe :

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
rate	Taux d'intérêt par période.
nper	Nombre total d'échéances de paiement dans une annuité.

Argument	Description
pmt	Paiement effectué à chaque échéance. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.
pv	Valeur, ou montant forfaitaire, que vaut une série de paiements à venir au moment présent. Si **pv** est omis, la fonction utilise 0 (zéro).
fv	Valeur future, ou solde de trésorerie, que vous souhaitez atteindre après le dernier paiement. Si **fv** est omis, la fonction utilise 0.
type	Doit être égal à 0 si les paiements sont dus à la fin de la période et égal à 1 si les paiements sont dus au début de la période. Si **type** est omis, la fonction utilise 0.

Exemples et résultats :

Exemple de script

Exemple	Résultat
Vous souhaitez vendre un équipement ménager en mensualités de $20. Le taux d'intérêt annuel est de 6 %. La facture arrive à la fin de chaque mois. Combien d'échéances sont nécessaires si le montant que vous devrez avoir touché une fois la dernière facture réglée est de $800 ? `nPer(0.005,-20,0,800)`	Renvoie 36.56.

Pmt

Cette fonction renvoie le paiement d'un emprunt sur la base de paiements périodiques constants et d'un taux d'intérêt constant. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise..

Pour obtenir le montant total versé sur toute la durée de l'emprunt, multipliez la valeur **pmt** renvoyée par **nper**.

Arguments :

Arguments

Argument	Description
rate	Taux d'intérêt par période.
nper	Nombre total d'échéances de paiement dans une annuité.
pv	Valeur, ou montant forfaitaire, que vaut une série de paiements à venir au moment présent. Si **pv** est omis, la fonction utilise 0 (zéro).

Argument	Description
fv	Valeur future, ou solde de trésorerie, que vous souhaitez atteindre après le dernier paiement. Si **fv** est omis, la fonction utilise 0.
type	Doit être égal à 0 si les paiements sont dus à la fin de la période et égal à 1 si les paiements sont dus au début de la période. Si **type** est omis, la fonction utilise 0.

Exemples et résultats :

Exemples de script

Exemple	Résultat
La formule suivante renvoie le paiement mensuel pour un emprunt de $20 000 selon un taux annuel de 10 %, qui doit être remboursé en 8 mois : `Pmt(0.1/12,8,20000)`	Renvoie -$2,594.66.
Pour le même emprunt, si le paiement est dû au début de la période, il correspond à : `Pmt(0.1/12,8,20000,0,1)`	Renvoie -$2,573.21.

PV

Cette fonction renvoie la valeur présente d'un investissement.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Type de données renvoyé : numérique Par défaut, le résultat sera formaté sous forme de devise..

La valeur présente correspond au montant total que vaut une série de paiements à venir au moment présent. Par exemple, pour un emprunt d'argent, le montant du prêt correspond à sa valeur présente pour le prêteur.

Arguments :

Arguments

Argument	Description
rate	Taux d'intérêt par période.
nper	Nombre total d'échéances de paiement dans une annuité.
pmt	Paiement effectué à chaque échéance. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.
fv	Valeur future, ou solde de trésorerie, que vous souhaitez atteindre après le dernier paiement. Si **fv** est omis, la fonction utilise 0.
type	Doit être égal à 0 si les paiements sont dus à la fin de la période et égal à 1 si les paiements sont dus au début de la période. Si **type** est omis, la fonction utilise 0.

Exemples et résultats :

Exemple de script

Exemple	Résultat
Quelle est la valeur présente d'une dette, lorsque vous devez verser $100 à la fin de chaque mois sur une période de cinq ans, avec un taux d'intérêt de 7 % ? `PV(0.07/12,12*5,-100,0,0)`	Renvoie $5,050.20.

Rate

Cette fonction renvoie le taux d'intérêt par période de l'annuité. Le résultat suit le format de nombre par défaut de la fonction **Fix** : deux décimales et en %.

Syntaxe :

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Type de données renvoyé : numérique

Le taux (**rate**) est calculé par itération et peut avoir zéro solution ou plus. Si les résultats successifs de **rate**ne concordent pas, une valeur NULL est renvoyée.

Arguments :

Arguments

Argument	Description
nper	Nombre total d'échéances de paiement dans une annuité.
pmt	Paiement effectué à chaque échéance. Il ne peut pas être modifié pendant la durée d'une annuité. Un paiement est défini sous la forme d'un nombre négatif, par exemple, -20.
pv	Valeur, ou montant forfaitaire, que vaut une série de paiements à venir au moment présent. Si **pv** est omis, la fonction utilise 0 (zéro).
fv	Valeur future, ou solde de trésorerie, que vous souhaitez atteindre après le dernier paiement. Si **fv** est omis, la fonction utilise 0.
type	Doit être égal à 0 si les paiements sont dus à la fin de la période et égal à 1 si les paiements sont dus au début de la période. Si **type** est omis, la fonction utilise 0.

Exemples et résultats :

Exemple de script

Exemple	Résultat
Quel est le taux d'intérêt d'un emprunt de $10 000 sur cinq ans avec des échéances mensuelles de $300 ? `Rate(60,-300,10000)`	Renvoie 2.00%.

5.12 Fonctions de formatage

Les fonctions de formatage déterminent le format d'affichage des expressions ou des champs numériques d'entrée. Selon le type de données, vous pouvez spécifier les caractères du séparateur décimal, du séparateur de milliers, etc.

Les fonctions renvoient toutes une valeur double comportant à la fois la chaîne et la valeur numérique, mais elles peuvent être considérées comme effectuant une conversion de nombre en chaîne. **Dual()** est un cas spécial, mais les autres fonctions de formatage utilisent la valeur numérique de l'expression d'entrée pour générer une chaîne représentant le nombre.

En revanche, les fonctions d'interprétation ont un comportement inverse : elles prennent les expressions de chaîne et les évaluent en tant que nombres, en spécifiant le format du nombre résultant.

Elles peuvent s'utiliser aussi bien dans les scripts de chargement de données que dans les expressions de graphique.

Toutes les représentations numériques sont données avec un point comme séparateur décimal.

Vue d'ensemble des fonctions de formatage

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

ApplyCodepage

ApplyCodepage() applique un jeu de caractères de page de codes différent au champ ou au texte spécifié dans l'expression. L'argument **codepage** doit être au format numérique.

```
ApplyCodepage (text, codepage)
```

Date

Date() formate une expression en tant que date en utilisant le format défini dans les variables système du script de chargement de données, sur le système d'exploitation ou dans une chaîne de format (si précisée).

```
Date (number[, format])
```

Dual

Dual() combine un nombre et une chaîne dans un même enregistrement, de sorte que la représentation numérique de l'enregistrement puisse servir à des fins de tri et de calcul tandis que la valeur de chaîne peut être utilisée à des fins d'affichage.

```
Dual (text, number)
```

Interval

Interval() formate un nombre en tant qu'intervalle de temps en utilisant le format défini dans les variables système du script de chargement de données, sur le système d'exploitation ou dans une chaîne de format (si précisée).

```
Interval (number[, format])
```

Money

Money() formate une expression numériquement sous forme de valeur monétaire, en utilisant le format défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée, et insère des séparateurs décimaux et de milliers facultatifs.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() formate un nombre, c'est-à-dire qu'il convertit la valeur numérique de l'entrée pour afficher un texte au format spécifié dans le deuxième paramètre. En cas d'omission du deuxième paramètre, il utilise les séparateurs de décimaux et de milliers définis dans le script de chargement de données. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() formate une expression en tant que valeur horaire en utilisant le format horaire défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

```
Time (number[, format])
```

Timestamp

TimeStamp() formate une expression en tant que valeur de date et heure en utilisant le format d'horodatage défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

```
Timestamp (number[, format])
```

Voir aussi :

p *Fonctions d'interprétation (page 1265)*

ApplyCodepage

ApplyCodepage() applique un jeu de caractères de page de codes différent au champ ou au texte spécifié dans l'expression. L'argument **codepage** doit être au format numérique.

Même s'il est possible d'utiliser ApplyCodepage dans les expressions de graphique, il est plus courant de l'employer comme fonction de script dans l'éditeur de chargement de données. Par exemple, comme vous chargez des fichiers qui ont pu être enregistrés dans des jeux de caractères différents hors de votre contrôle, vous pouvez appliquer la page de codes qui représente le jeu de caractères requis.

Syntaxe :

```
ApplyCodepage(text, codepage)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Champ ou texte auquel vous souhaitez appliquer une page de codes différente, que l'argument **codepage** fournit.
codepage	Nombre représentant la page de codes à appliquer au champ ou à l'expression fourni(e) par **text**.

Exemples et résultats :

Exemples de script

Exemple	Résultat
LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;	Lors du chargement à partir de SQL, la source peut comporter un mélange de jeux de caractères différents : cyrillique, hébreu, et ainsi de suite, à partir du format UTF-8. Il est alors nécessaire de charger ces jeux ligne par ligne, en appliquant une page de codes différente à chaque ligne. La valeur de **codepage** 1253 représente le jeu de caractères grec Windows, la valeur 1255 l'hébreu et la valeur 65001 les caractères UTF-8 latins standard.

Voir aussi : *Jeu de caractères (page 162)*

Date

Date() formate une expression en tant que date en utilisant le format défini dans les variables système du script de chargement de données, sur le système d'exploitation ou dans une chaîne de format (si précisée).

Syntaxe :

```
Date(number[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne décrivant le format de la chaîne résultante. Si aucune chaîne de format n'est fournie, le format de date défini dans les variables système du script de chargement de données ou du système d'exploitation est utilisé.

Exemples et résultats :
Soient les paramètres par défaut suivants dans les exemples ci-dessous :

- Paramètre de date 1 : YY-MM-DD
- Paramètre de date 2 : M/D/YY

Exemple :

```
Date( A )
```
où A=35648

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	97-08-06	8/6/97
Nombre :	35648	35648

Exemple :

```
Date( A, 'YY.MM.DD' )
```
où A=35648

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	97.08.06	97.08.06
Nombre :	35648	35648

Exemple :

```
Date( A, 'DD.MM.YYYY' )
```
où A=35648.375

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	06.08.1997	06.08.1997
Nombre :	35648.375	35648.375

Exemple :

```
Date( A, 'YY.MM.DD' )
```
où A=8/6/97

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	NULL (rien)	97.08.06
Nombre :	NULL	35648

Dual

Dual() combine un nombre et une chaîne dans un même enregistrement, de sorte que la représentation numérique de l'enregistrement puisse servir à des fins de tri et de calcul tandis que la valeur de chaîne peut être utilisée à des fins d'affichage.

Syntaxe :

```
Dual(text, number)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Valeur de chaîne à utiliser en combinaison avec l'argument du nombre.
number	Nombre à utiliser en combinaison avec la chaîne dans l'argument de la chaîne.

Dans Qlik Sense, toutes les valeurs de champs sont potentiellement des valeurs doubles. Autrement dit, les valeurs de champ peuvent comporter à la fois une valeur numérique et une valeur textuelle. Prenons l'exemple d'une date, qui peut admettre une valeur numérique de 40908 et une représentation textuelle égale à '2011-12-31'.

Lorsque plusieurs éléments de données lus dans un champ sont dotés de représentations de chaîne différentes mais qu'ils disposent de la même représentation numérique valide, ils partagent tous la première représentation de chaîne rencontrée.

*La fonction **dual** est généralement utilisée au début du script, avant que d'autres données ne soient lues dans le champ concerné, afin de créer cette première représentation de chaîne qui sera affichée dans les volets de filtre.*

Exemples et résultats :

Exemples de script

Exemple	Description
Ajoutez les exemples suivants à votre script et exécutez ce dernier. `Load dual ( NameDay,NumDay ) as DayOfWeek inline [ NameDay,NumDay Monday,0 Tuesday,1 Wednesday,2 Thursday,3 Friday,4 Saturday,5 Sunday,6 ];`	Il est possible d'utiliser le champ DayOfWeek dans une visualisation, comme dimension, par exemple. Ainsi, dans une table, les jours de la semaine sont triés automatiquement selon leur séquence numérique correcte plutôt que par ordre alphabétique.
`Load Dual('Q' & Ceil(Month(Now ())/3), Ceil(Month (Now())/3)) as Quarter AutoGenerate 1;`	Cet exemple identifie le trimestre approprié. Il s'affiche sous l'intitulé Q1 lorsque la fonction **Now()** est exécutée pendant les trois premiers mois de l'année, sous l'intitulé Q2 pendant les trois mois suivants, et ainsi de suite. Cependant, lorsqu'il est utilisé pour le tri, le champ Quarter (Trimestre) se comporte comme sa valeur numérique : 1 à 4.
`Dual('Q' & Ceil (Month(Date)/3), Ceil(Month (Date)/3)) as Quarter`	À l'instar de l'exemple précédent, le champ Quarter est créé à l'aide des valeurs de texte 'Q1' à 'Q4', et se voit attribuer les valeurs numériques 1 à 4. Afin de pouvoir appliquer cela dans le script, vous devez charger les valeurs de Date.
`Dual(WeekYear (Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Cet exemple crée un champ YearWeek comportant des valeurs de texte de la forme '2012-W22' et, dans le même temps, attribue une valeur numérique correspondant au numéro de date du premier jour de la semaine, par exemple : 41057. Afin de pouvoir appliquer cela dans le script, vous devez charger les valeurs de Date.

Interval

Interval() formate un nombre en tant qu'intervalle de temps en utilisant le format défini dans les variables système du script de chargement de données, sur le système d'exploitation ou dans une chaîne de format (si précisée).

Les intervalles peuvent prendre la forme d'heures, de jours ou d'une combinaison de jours, d'heures, de minutes, de secondes et de fractions de seconde.

Syntaxe :

```
Interval(number[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne décrivant la façon dont la chaîne d'intervalle résultante doit être formatée. En cas d'omission, ce sont le format de date abrégé, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés.

Exemples et résultats :
Soient les paramètres par défaut suivants dans les exemples ci-dessous :

- Paramètre de format de date 1 : YY-MM-DD
- Paramètre de format de date 2 : hh:mm:ss
- Séparateur décimal des nombres : .

Table des résultats

Exemple	Chaîne	Nombre
`Interval( A )` où A=0.375	09:00:00	0.375
`Interval( A )` où A=1.375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` où A=1.375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` où A=97-08-06 09:00:00 et B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() formate une expression numériquement sous forme de valeur monétaire, en utilisant le format défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée, et insère des séparateurs décimaux et de milliers facultatifs.

Syntaxe :

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne décrivant la façon dont la chaîne monétaire résultante doit être formatée.
dec_sep	Chaîne indiquant le séparateur de nombres décimaux.
thou_sep	Chaîne indiquant le séparateur de milliers.

Si les arguments 2-4 sont omis, c'est le format de la devise défini dans le système d'exploitation qui est utilisé.

Exemples et résultats :
Soient les paramètres par défaut suivants dans les exemples ci-dessous :

- Paramètre de format monétaire 1 : kr ##0,00, MoneyThousandSep' '
- Paramètre de format monétaire 2 : $ #,##0.00, MoneyThousandSep','

Exemple :

```
Money( A )
```
où A=35648

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	kr 35 648,00	$ 35,648.00
Nombre :	35648.00	35648.00

Exemple :

```
Money( A, '#,##0 ¥', '.' , ',' )
```
où A=3564800

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	3,564,800 ¥	3,564,800 ¥
Nombre :	3564800	3564800

Num

Num() formate un nombre, c'est-à-dire qu'il convertit la valeur numérique de l'entrée pour afficher un texte au format spécifié dans le deuxième paramètre. En cas d'omission du deuxième paramètre, il utilise les séparateurs de décimaux et de milliers définis dans le script de chargement de données. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

Syntaxe :

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Type de données renvoyé : double

La fonction Num renvoie une valeur double comportant à la fois la chaîne et la valeur numérique. La fonction prend la valeur numérique de l'expression entrée et génère une chaîne représentant le nombre.

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne spécifiant la façon dont la chaîne résultante doit être formatée. En cas d'omission, les séparateurs de décimaux et de milliers définis dans le script de chargement de données sont utilisés.
dec_sep	Chaîne indiquant le séparateur de nombres décimaux. En cas d'omission, la valeur de la variable DecimalSep définie dans le script de chargement de données est utilisée.
thou_sep	Chaîne indiquant le séparateur de milliers. En cas d'omission, la valeur de la variable ThousandSep définie dans le script de chargement de données est utilisée.

Exemple : Expression de graphique

Exemple :

La table suivante montre les résultats lorsque le champ A est égal à 35648.312.

Résultats

La présence d'une icône	Résultat
Num(A)	35648.312 (dépend des variables d'environnement du script)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Exemple : Script de chargement

Script de chargement

Vous pouvez utiliser *Num* dans un script de chargement afin de formater un nombre, même si le séparateur de milliers et le séparateur décimal sont déjà définis dans le script. Le script de chargement ci-dessous comprend un séparateur décimal et un séparateur de milliers spécifiques, mais utilise cependant *Num* pour formater les données de différentes façons.

Dans l'**éditeur de chargement de données**, créez une section, puis ajoutez et exécutez l'exemple de script. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de votre application afin de visualiser le résultat.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(transaction_amount) as [No formatting], Num(transaction_amount,'0') as [0], Num(transaction_amount,'#,##0') as [#,##0], Num(transaction_amount,'# ###,00') as [# ###,00], Num(transaction_amount,'# ###,00',',',' ') as [# ###,00 , ',' , ' '], Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' , ','], Num(transaction_amount,'$#,###.00') as [$#,###.00], ; Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23, 0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757, 20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Table Qlik Sense affichant les résultats de différentes utilisations de la fonction *Num* dans le script de chargement. À titre d'exemple, la quatrième colonne de la table contient une utilisation de formatage incorrecte.

Aucun formatage	0	#,##0	# ###,00	# ###,00 , ',' , ''	#,###.00 , '.' , ','	$#,###.00
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

Exemple : Script de chargement

Script de chargement

Vous pouvez utiliser *Num* dans un script de chargement afin de formater un nombre sous forme de pourcentage.

Dans l'**éditeur de chargement de données**, créez une section, puis ajoutez et exécutez l'exemple de script. Ensuite, ajoutez au moins les champs répertoriés dans la colonne des résultats à une feuille de votre application afin de visualiser le résultat.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(discount,'#,##0%') as
[Discount #,##0%] ; Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23,
0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1,
0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21,
1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757,
20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Table Qlik Sense affichant les résultats de la fonction *Num* utilisée dans le script de chargement en vue de formater les pourcentages.

Remise	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() formate une expression en tant que valeur horaire en utilisant le format horaire défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

Syntaxe :

```
Time(number[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne décrivant la façon dont la chaîne horaire résultante doit être formatée. En cas d'omission, ce sont le format de date court, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés.

Exemples et résultats :
Soient les paramètres par défaut suivants dans les exemples ci-dessous :

- Paramètre de format horaire 1 : hh:mm:ss
- Paramètre de format horaire 2 : hh.mm.ss

Exemple :

```
Time( A )
```
où A=0.375

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	09:00:00	09.00.00
Nombre :	0.375	0.375

Exemple :

```
Time( A )
```
où A=35648.375

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	09:00:00	09.00.00
Nombre :	35648.375	35648.375

Exemple :

```
Time( A, 'hh-mm' )
```
où A=0.99999

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	23-59	23-59
Nombre :	0.99999	0.99999

Timestamp

TimeStamp() formate une expression en tant que valeur de date et heure en utilisant le format d'horodatage défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

Syntaxe :

```
Timestamp(number[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
number	Nombre à formater.
format	Chaîne décrivant la façon dont la chaîne d'horodatage résultante doit être formatée. En cas d'omission, ce sont le format de date court, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés.

Exemples et résultats :
Soient les paramètres par défaut suivants dans les exemples ci-dessous :

- Paramètre TimeStampFormat 1 : YY-MM-DD hh:mm:ss
- Paramètre TimeStampFormat 2 : M/D/YY hh:mm:ss

Exemple :

```
Timestamp( A )
```
où A=35648.375

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	97-08-06 09:00:00	8/6/97 09:00:00
Nombre :	35648.375	35648.375

Exemple :

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
où A=35648

Table des résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	1997-08-06 00.00	1997-08-06 00.00
Nombre :	35648	35648

5.13 Fonctions numériques générales

Dans ces fonctions numériques générales, les arguments sont des expressions où **x** doit être interprété comme un nombre réel. Elles s'utilisent toutes aussi bien dans les scripts de chargement de données que dans les expressions de graphique.

Vue d'ensemble des fonctions numériques générales

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

bitcount

BitCount() renvoie le nombre de bits définis sur 1 dans l'équivalent binaire d'un nombre décimal. Cela signifie que la fonction renvoie le nombre de bits définis dans **integer_number**, où **integer_number** est interprété comme un entier de 32 bits signé.

```
BitCount(integer_number)
```

div

Div() renvoie la partie entière de la division arithmétique du premier argument par le second. Les deux paramètres sont interprétés comme des nombres réels, c'est-à-dire que ce ne sont pas nécessairement des entiers.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() renvoie la valeur absolue de **x**. Le résultat est un nombre positif.

```
Fabs (x)
```

fact

Fact() renvoie la factorielle d'un entier positif **x**.

```
Fact (x)
```

frac

Frac() renvoie la partie fractionnaire de **x**.

```
Frac (x)
```

sign

Sign() renvoie 1, 0 ou -1 selon que **x** est un nombre positif, 0 ou un nombre négatif.

```
Sign (x)
```

Fonctions de combinaison et de permutation

combin

Combin() renvoie le nombre de combinaisons de **q** éléments sélectionnables dans un ensemble de **p** éléments. Comme représenté par la formule : `Combin(p,q) = p! / q!(p-q)!` L'ordre dans lequel les éléments sont sélectionnés n'a pas d'importance.

```
Combin (p, q)
```

permut

Permut() renvoie le nombre de permutations de **q** éléments sélectionnables dans un groupe de **p** éléments. Comme représenté par la formule : `Permut(p,q) = (p)! / (p - q)!` L'ordre dans lequel les éléments sont sélectionnés a de l'importance.

```
Permut (p, q)
```

Fonctions modulo

fmod

fmod() est une fonction modulo généralisée qui renvoie le reste de la division entière du premier argument (le dividende) par le second (le diviseur). Le résultat est un nombre réel. Les deux arguments sont interprétés comme des nombres réels, c'est-à-dire que ce ne sont pas nécessairement des entiers.

```
Fmod (a, b)
```

mod

Mod() est une fonction modulo mathématique qui renvoie le reste non négatif d'une division entière. Le premier argument est le dividende, le second le diviseur. Tous deux doivent être des valeurs entières.

```
Mod (integer_number1, integer_number2)
```

Fonctions de parité

even

Even() renvoie True (-1) si **integer_number** est un entier pair ou est égal à zéro. Il renvoie False (0) si **integer_number** est un entier impair et NULL si **integer_number** n'est pas un entier.

```
Even (integer_number)
```

odd

Odd() renvoie True (-1) si **integer_number** est un entier impair ou est égal à zéro. Il renvoie False (0) si **integer_number** est un entier pair et NULL si **integer_number** n'est pas un entier.

```
Odd (integer_number)
```

Fonction d'arrondi

ceil

Ceil() arrondit une valeur au multiple supérieur le plus proche du pas **step** décalé du nombre **offset** défini.

```
Ceil (x[, step[, offset]])
```

floor

Floor() arrondit une valeur au multiple inférieur le plus proche du pas **step** décalé du nombre **offset** défini.

```
Floor (x[, step[, offset]])
```

round

Round() renvoie le résultat de l'arrondissement d'un nombre au multiple supérieur ou inférieur le plus proche du pas **step** décalé du nombre **offset** défini.

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() renvoie le nombre de bits définis sur 1 dans l'équivalent binaire d'un nombre décimal. Cela signifie que la fonction renvoie le nombre de bits définis dans **integer_number**, où **integer_number** est interprété comme un entier de 32 bits signé.

Syntaxe :

```
BitCount(integer_number)
```

Type de données renvoyé : entier

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`BitCount ( 3 )`	3 correspondant à 11 en binaire, 2 est donc renvoyé.
`BitCount ( -1 )`	-1 correspondant à 64 uns en binaire, 64 est donc renvoyé.

Ceil

Ceil() arrondit une valeur au multiple supérieur le plus proche du pas **step** décalé du nombre **offset** défini.

Comparez ces résultats à ceux de la fonction **floor**, qui arrondit les nombres vers la valeur inférieure la plus proche.

Syntaxe :

```
Ceil(x[, step[, offset]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
x	Nombre entré.
step	Incrément de l'intervalle. La valeur par défaut est de 1.
offset	Définit la base de l'intervalle du pas. La valeur par défaut est de 0.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Ceil(2.4 )`	Renvoie 3. Dans cet exemple, la taille du pas correspond à 1 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Renvoie 5.
`Ceil(3.88 ,0.1)`	Renvoie 3.9. Dans cet exemple, la taille de l'intervalle correspond à 0.1 et la base de l'intervalle à 0. Les intervalles sont les suivants : ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Renvoie 5.
`Ceil(1.1 ,1)`	Renvoie 2.
`Ceil(1.1 ,1,0.5)`	Renvoie 1.5. Dans cet exemple, la taille du pas correspond à 1 et le décalage à 0.5. Cela signifie que la base de l'intervalle du pas correspond à 0.5 et non à 0. Les intervalles sont les suivants : ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...

Exemples	Résultats
`Ceil(1.1 ,1,-0.01)`	Renvoie 1.99. Les intervalles sont les suivants : ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() renvoie le nombre de combinaisons de **q** éléments sélectionnables dans un ensemble de **p** éléments. Comme représenté par la formule : `Combin(p,q) = p! / q!(p-q)!` L'ordre dans lequel les éléments sont sélectionnés n'a pas d'importance.

Syntaxe :

```
Combin(p, q)
```

Type de données renvoyé : entier

Limitations :

Les éléments non entiers seront tronqués.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
Combien de combinaisons de 7 nombres est-il possible de tirer d'un total de 35 numéros de loto ? `Combin( 35,7 )`	Renvoie 6 724 520.

Div

Div() renvoie la partie entière de la division arithmétique du premier argument par le second. Les deux paramètres sont interprétés comme des nombres réels, c'est-à-dire que ce ne sont pas nécessairement des entiers.

Syntaxe :

```
Div(integer_number1, integer_number2)
```

Type de données renvoyé : entier

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Div( 7,2 )`	Renvoie 3.
`Div( 7.1,2.3 )`	Renvoie 3.
`Div( 9,3 )`	Renvoie 3.
`Div( -4,3 )`	Renvoie -1.
`Div( 4,-3 )`	Renvoie -1.
`Div( -4,-3 )`	Renvoie 1.

Even

Even() renvoie True (-1) si **integer_number** est un entier pair ou est égal à zéro. Il renvoie False (0) si **integer_number** est un entier impair et NULL si **integer_number** n'est pas un entier.

Syntaxe :

```
Even(integer_number)
```

Type de données renvoyé : booléen

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Even( 3 )`	Renvoie 0, False.
`Even( 2 * 10 )`	Renvoie -1, True.
`Even( 3.14 )`	Renvoie NULL.

Fabs

Fabs() renvoie la valeur absolue de **x**. Le résultat est un nombre positif.

Syntaxe :

```
fabs(x)
```

Type de données renvoyé : numérique

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`fabs( 2.4 )`	Renvoie 2.4.
`fabs( -3.8 )`	Renvoie 3.8.

Fact

Fact() renvoie la factorielle d'un entier positif **x**.

Syntaxe :

```
Fact(x)
```

Type de données renvoyé : entier

Limitations :

Si le nombre **x** n'est pas un entier, il sera tronqué. Les nombres non positifs renvoient la valeur NULL.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Fact( 1 )`	Renvoie 1.
`Fact( 5 )`	Renvoie 120 (1 * 2 * 3 * 4 * 5 = 120).
`Fact( -5 )`	Renvoie NULL.

Floor

Floor() arrondit une valeur au multiple inférieur le plus proche du pas **step** décalé du nombre **offset** défini.

Comparez ces résultats à ceux de la fonction **ceil**, qui arrondit les nombres à la valeur supérieure la plus proche.

Syntaxe :

```
Floor(x[, step[, offset]])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
x	Nombre entré.
step	Incrément de l'intervalle. La valeur par défaut est de 1.
offset	Définit la base de l'intervalle du pas. La valeur par défaut est de 0.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Floor(2.4)`	Renvoie 2. In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Renvoie 4.
`Floor(3.88 ,0.1)`	Renvoie 3.8. Dans cet exemple, la taille de l'intervalle correspond à 0.1 et la base de l'intervalle à 0. Les intervalles sont les suivants : ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Renvoie 0.
`Floor(1.1 ,1)`	Renvoie 1.
`Floor(1.1 ,1,0.5)`	Renvoie 0.5. Dans cet exemple, la taille du pas correspond à 1 et le décalage à 0.5. Cela signifie que la base de l'intervalle du pas correspond à 0.5 et non à 0. Les intervalles sont les suivants : ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() est une fonction modulo généralisée qui renvoie le reste de la division entière du premier argument (le dividende) par le second (le diviseur). Le résultat est un nombre réel. Les deux arguments sont interprétés comme des nombres réels, c'est-à-dire que ce ne sont pas nécessairement des entiers.

Syntaxe :

```
fmod(a, b)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
a	Dividende
b	Diviseur

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`fmod( 7,2 )`	Renvoie 1.
`fmod( 7.5,2 )`	Renvoie 1.5.
`fmod( 9,3 )`	Renvoie 0.
`fmod( -4,3 )`	Renvoie -1.
`fmod( 4,-3 )`	Renvoie 1.
`fmod( -4,-3 )`	Renvoie -1.

Frac

Frac() renvoie la partie fractionnaire de **x**.

La fraction est définie de telle façon que `Frac(x ) + Floor(x ) = x`. Plus simplement, cela signifie que la partie fractionnaire d'un nombre positif correspond à la différence entre le nombre (x) et l'entier qui précède la partie fractionnaire.

Par exemple : la partie fractionnaire de 11.43 = 11.43 - 11 = 0.43

Pour un nombre négatif, soit -1.4, Floor(-1.4) = -2, ce qui génère le résultat suivant :

La partie fractionnaire de -1.4 = 1.4 - (-2) = -1.4 + 2 = 0.6

Syntaxe :

```
Frac(x)
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
x	Nombre pour lequel la fraction est à renvoyer.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Frac( 11.43 )`	Renvoie 0.43.
`Frac( -1.4 )`	Renvoie 0.6.
Extrait la composante heure de la représentation numérique d'un horodatage, omettant ainsi la date. `Time(Frac(44518.663888889))`	Renvoie 3:56:00 PM

Mod

Mod() est une fonction modulo mathématique qui renvoie le reste non négatif d'une division entière. Le premier argument est le dividende, le second le diviseur. Tous deux doivent être des valeurs entières.

Syntaxe :

```
Mod(integer_number1, integer_number2)
```

Type de données renvoyé : entier

Limitations :

integer_number2 doit être supérieur à 0.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Mod( 7,2 )`	Renvoie 1.
`Mod( 7.5,2 )`	Renvoie NULL.
`Mod( 9,3 )`	Renvoie 0.

Exemples	Résultats
`Mod( -4,3 )`	Renvoie 2.
`Mod( 4,-3 )`	Renvoie NULL.
`Mod( -4,-3 )`	Renvoie NULL.

Odd

Odd() renvoie True (-1) si **integer_number** est un entier impair ou est égal à zéro. Il renvoie False (0) si **integer_number** est un entier pair et NULL si **integer_number** n'est pas un entier.

Syntaxe :

```
Odd(integer_number)
```

Type de données renvoyé : booléen

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Odd( 3 )`	Renvoie -1, True.
`Odd( 2 * 10 )`	Renvoie 0, False.
`Odd( 3.14 )`	Renvoie NULL.

Permut

Permut() renvoie le nombre de permutations de **q** éléments sélectionnables dans un groupe de **p** éléments. Comme représenté par la formule : `Permut(p,q) = (p)! / (p - q)!` L'ordre dans lequel les éléments sont sélectionnés a de l'importance.

Syntaxe :

```
Permut(p, q)
```

Type de données renvoyé : entier

Limitations :

Les arguments non entiers seront tronqués.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
De combien de façons les médailles d'or, d'argent et de bronze pourraient-elles être distribuées après une finale du 100 m comptant 8 participants ? `Permut( 8,3 )`	Renvoie 336.

Round

Round() renvoie le résultat de l'arrondissement d'un nombre au multiple supérieur ou inférieur le plus proche du pas **step** décalé du nombre **offset** défini.

Si le nombre à arrondir se trouve exactement au milieu d'un intervalle, il est arrondi à la valeur supérieure la plus proche.

Syntaxe :

```
Round(x[, step[, offset]])
```

Type de données renvoyé : numérique

Si vous arrondissez un nombre à virgule flottante, vous risquez d'obtenir des résultats erronés. Ces erreurs d'arrondissement s'expliquent par le fait que les nombres à virgule flottante sont représentés par un nombre fini de chiffres binaires. De ce fait, les résultats sont calculés à l'aide d'un nombre qui est déjà arrondi. Si ces erreurs d'arrondissement doivent influer sur votre travail, multipliez les nombres afin de les convertir en entiers avant de les arrondir.

Arguments :

Arguments

Argument	Description
x	Nombre entré.
step	Incrément de l'intervalle. La valeur par défaut est de 1.
offset	Définit la base de l'intervalle du pas. La valeur par défaut est de 0.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Round(3.8 )`	Renvoie 4. Dans cet exemple, la taille du pas correspond à 1 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Renvoie 4.
`Round(2.5 )`	Renvoie 3. Dans cet exemple, la taille du pas correspond à 1 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Renvoie 4. Résultat arrondi, car 2 correspond exactement à la moitié de l'intervalle de pas 4. Dans cet exemple, la taille du pas correspond à 4 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Renvoie 0. Résultat arrondi, car 2 est inférieur à la moitié de l'intervalle de pas 6. Dans cet exemple, la taille du pas correspond à 6 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Renvoie 3.9. Dans cet exemple, la taille du pas correspond à 0,1 et la base de l'intervalle du pas à 0. Les intervalles sont les suivants : ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Renvoie 3.889 Dans cet exemple, la taille du pas est de 0.001, ce qui arrondit le nombre à la valeur supérieure et le limite à trois décimales.
`Round(3.88 ,5)`	Renvoie 5.
`Round(1.1 ,1,0.5)`	Renvoie 1.5. Dans cet exemple, la taille du pas correspond à 1 et la base de l'intervalle du pas à 0,5. Les intervalles sont les suivants : ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() renvoie 1, 0 ou -1 selon que **x** est un nombre positif, 0 ou un nombre négatif.

Syntaxe :

```
Sign(x)
```

Type de données renvoyé : numérique

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples et résultats

Exemples	Résultats
`Sign( 66 )`	Renvoie 1.
`Sign( 0 )`	Renvoie 0.
`Sign( - 234 )`	Renvoie -1.

5.14 Fonctions géospatiales

Ces fonctions permettent de gérer les données géospatiales dans les visualisations de carte. Qlik Sense suit les spécifications GeoJSON pour les données géospatiales et prend en charge les types de géométrie suivants :

- Point
- Linestring
- Polygon
- Multipolygon

Pour plus d'informations sur les spécifications GeoJSON, voir :
≤ GeoJSON.org

Vue d'ensemble des fonctions géospatiales

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Il existe deux catégories de fonctions géospatiales : les fonctions d'agrégation et les fonctions de non-agrégation.

Les fonctions d'agrégation utilisent un ensemble d'éléments géométriques (points ou zones) comme données d'entrée et renvoient une géométrie unique. Par exemple, il est possible de fusionner ensemble plusieurs zones et de tracer sur la carte une délimitation unique pour l'agrégation.

Les fonctions de non-agrégation utilisent une géométrie unique et renvoient une seule géométrie. Par exemple, pour la fonction GeoGetPolygonCenter(), si la géométrie de délimitation d'une zone est définie comme données d'entrée, la géométrie des points (longitude et latitude) du centre de cette zone est renvoyée.

Les fonctions suivantes sont des fonctions d'agrégation :

GeoAggrGeometry

GeoAggrGeometry() s'utilise pour agréger plusieurs zones en une zone plus grande, par exemple des sous-régions agrégées en une région.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() s'utilise pour agréger une géométrie dans une zone et calculer le plus petit cadre de délimitation contenant l'ensemble des coordonnées.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() s'utilise pour déterminer le nombre de sommets contenus dans une géométrie de type polygone.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() s'utilise pour agréger une géométrie dans une zone et appliquer l'inverse d'une projection.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() s'utilise pour agréger une géométrie dans une zone et appliquer une projection.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() permet de réduire le nombre de sommets d'une géométrie et d'agréger plusieurs zones en une seule, tout en continuant à afficher les lignes de délimitation des différentes zones.

```
GeoReduceGeometry (geometry)
```

Les fonctions suivantes sont des fonctions de non-agrégation :

GeoGetBoundingBox

GeoGetBoundingBox() s'utilise dans les scripts et les expressions de graphique pour calculer le plus petit cadre de délimitation géospatial contenant l'ensemble des coordonnées d'une géométrie.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() s'utilise dans les scripts et les expressions de graphique pour calculer et renvoyer le point central d'une géométrie.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() s'utilise dans les scripts et les expressions de graphique pour créer et repérer un point par une balise de latitude et longitude.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() s'utilise dans les scripts et les expressions de graphique pour appliquer une projection à une géométrie.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() s'utilise pour agréger plusieurs zones en une zone plus grande, par exemple des sous-régions agrégées en une région.

Syntaxe :

```
GeoAggrGeometry(field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

En général, **GeoAggrGeometry()** s'utilise pour combiner des données de délimitation géospatiales. Par exemple, il se peut que vous disposiez de zones de codes postaux pour les banlieues d'une grande ville et du chiffre d'affaires associé à chaque zone. Si le territoire d'un vendeur couvre plusieurs zones de codes postaux, il peut s'avérer utile de présenter les ventes totales par territoire commercial plutôt que par zone individuelle et d'afficher les résultats sur une carte en couleur.

GeoAggrGeometry() permet de calculer l'agrégation des différentes géométries de banlieues et de générer la géométrie fusionnée du territoire dans le modèle de données. Ensuite, après ajustement des délimitations du territoire commercial, lorsque les données sont rechargées, les nouvelles délimitations et chiffres d'affaires fusionnés sont reflétés sur la carte.

Comme **GeoAggrGeometry()** est une fonction d'agrégation, si vous l'utilisez dans le script, vous devez disposer d'une instruction **LOAD** accompagnée d'une clause **Group by**.

*Les lignes de délimitation des cartes créées à l'aide de **GeoAggrGeometry()** correspondent à celles des zones fusionnées. Si vous souhaitez afficher les lignes de délimitation individuelles des zones avant agrégation, utilisez **GeoReduceGeometry()**.*

Exemples :
Dans cet exemple, sont chargés tour à tour un fichier KML comportant des données de zones, puis une table comprenant les données de zones agrégées.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoAggrGeometry(world.Area) as
[AggrArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() s'utilise pour agréger une géométrie dans une zone et calculer le plus petit cadre de délimitation contenant l'ensemble des coordonnées.

Une fonction GeoBoundingBox est représentée sous la forme d'une liste de quatre valeurs : gauche, droite, haut, bas.

Syntaxe :

```
GeoBoundingBox(field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

GeoBoundingBox() agrège un ensemble de géométries et renvoie quatre coordonnées pour le plus petit rectangle contenant toutes les coordonnées de cette géométrie agrégée.

Pour visualiser le résultat sur une carte, transférez la chaîne de quatre coordonnées obtenue dans un format de polygone, balisez le champ transféré à l'aide d'un format de géopolygone, puis faites glisser ce champ et déposez-le dans l'objet carte. Les zones rectangulaires s'afficheront ensuite dans la visualisation de la carte.

GeoCountVertex

GeoCountVertex() s'utilise pour déterminer le nombre de sommets contenus dans une géométrie de type polygone.

Syntaxe :

```
GeoCountVertex(field_name)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

GeoGetBoundingBox

GeoGetBoundingBox() s'utilise dans les scripts et les expressions de graphique pour calculer le plus petit cadre de délimitation géospatial contenant l'ensemble des coordonnées d'une géométrie.

Un cadre de délimitation géospatial, créé par la fonction GeoBoundingBox(), est représenté sous la forme d'une liste de quatre valeurs : gauche, droite, haut, bas.

Syntaxe :

```
GeoGetBoundingBox(field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

*N'utilisez pas la clause **Group by** dans l'éditeur de chargement de données avec cette fonction et les autres fonctions géospatiales qui n'agrègent pas, car elle provoquerait une erreur au moment du chargement.*

GeoGetPolygonCenter

GeoGetPolygonCenter() s'utilise dans les scripts et les expressions de graphique pour calculer et renvoyer le point central d'une géométrie.

Dans certains cas, il est nécessaire de tracer un point au lieu de colorer une zone sur une carte. Si les données géospatiales existantes ne sont disponibles que sous la forme d'une géométrie de zone (par exemple, une délimitation), faites appel à la fonction **GeoGetPolygonCenter()** pour récupérer une paire de coordonnées de longitude et de latitude afin d'identifier le centre de la zone.

Syntaxe :

```
GeoGetPolygonCenter(field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

*N'utilisez pas la clause **Group by** dans l'éditeur de chargement de données avec cette fonction et les autres fonctions géospatiales qui n'agrègent pas, car elle provoquerait une erreur au moment du chargement.*

GeoInvProjectGeometry

GeoInvProjectGeometry() s'utilise pour agréger une géométrie dans une zone et appliquer l'inverse d'une projection.

Syntaxe :

```
GeoInvProjectGeometry(type, field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
type	Type de projection utilisé lors de la transformation de la géométrie de la carte. Cet argument admet l'une de ces deux valeurs : 'unit' (par défaut), qui produit une projection 1:1, ou 'mercator', qui utilise la projection de Mercator standard.
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

Exemple :

Exemple de script

Exemple	Résultat
Dans une instruction `Load` : `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	La géométrie chargée en tant que **AreaPolygon** est transformée à l'aide de la transformation inverse de la projection de Mercator et stockée en tant que **InvProjectGeometry** à des fins d'utilisation dans les visualisations.

GeoMakePoint

GeoMakePoint() s'utilise dans les scripts et les expressions de graphique pour créer et repérer un point par une balise de latitude et longitude. GeoMakePoint renvoie les points dans l'ordre de la longitude et de la latitude.

Syntaxe :

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Type de données renvoyé : chaîne, formatée [longitude, latitude]

Arguments :

Arguments

Argument	Description
lat_field_name	Champ ou expression faisant référence à un champ représentant la latitude du point.
lon_field_ name	Champ ou expression faisant référence à un champ représentant la longitude du point.

*N'utilisez pas la clause **Group by** dans l'éditeur de chargement de données avec cette fonction et les autres fonctions géospatiales qui n'agrègent pas, car elle provoquerait une erreur au moment du chargement.*

GeoProject

GeoProject() s'utilise dans les scripts et les expressions de graphique pour appliquer une projection à une géométrie.

Syntaxe :

```
GeoProject(type, field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
type	Type de projection utilisé lors de la transformation de la géométrie de la carte. Cet argument admet l'une de ces deux valeurs : 'unit' (par défaut), qui produit une projection 1:1, ou 'mercator', qui utilise la projection de Mercator Web.
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

*N'utilisez pas la clause **Group by** dans l'éditeur de chargement de données avec cette fonction et les autres fonctions géospatiales qui n'agrègent pas, car elle provoquerait une erreur au moment du chargement.*

Exemple :

Exemples de script

Exemple	Résultat
Dans une instruction `Load` : `GeoProject ('mercator',Area) as GetProject`	La projection de Mercator est appliquée à la géométrie chargée en tant que **Area** et le résultat est stocké sous la forme **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() s'utilise pour agréger une géométrie dans une zone et appliquer une projection.

Syntaxe :

```
GeoProjectGeometry(type, field_name)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
type	Type de projection utilisé lors de la transformation de la géométrie de la carte. Cet argument admet l'une de ces deux valeurs : 'unit' (par défaut), qui produit une projection 1:1, ou 'mercator', qui utilise la projection de Mercator Web.
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

Exemple :

Exemple	Résultat
Dans une instruction `Load` : `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	La géométrie chargée en tant que **AreaPolygon** est transformée à l'aide de la projection de Mercator et stockée en tant que **ProjectGeometry** à des fins d'utilisation dans les visualisations.

GeoReduceGeometry

GeoReduceGeometry() permet de réduire le nombre de sommets d'une géométrie et d'agréger plusieurs zones en une seule, tout en continuant à afficher les lignes de délimitation des différentes zones.

Syntaxe :

```
GeoReduceGeometry(field_name[, value])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
field_name	Champ ou expression faisant référence à un champ contenant la géométrie à représenter. Il peut s'agir soit d'un point (ou d'un ensemble de points) indiquant la longitude et la latitude, soit d'une zone.

Argument	Description
value	Quantité de réduction à appliquer à la géométrie. La plage de valeurs est comprise entre 0 et 1, 0 représentant l'absence de réduction et 1, la réduction maximale des sommets. *L'emploi d'un argument value de 0.9 ou plus avec un ensemble de données complexe peut réduire le nombre de sommets à un niveau où la représentation visuelle devient inexacte.*

La fonction **GeoReduceGeometry()** exécute une fonction semblable à **GeoAggrGeometry()** dans le sens où elle agrège un nombre de zones en une zone unique. La différence réside dans le fait que les lignes de délimitation individuelles issues des données préalables à l'agrégation sont affichées sur la carte si vous utilisez **GeoReduceGeometry()**.

Comme **GeoReduceGeometry()** est une fonction d'agrégation, si vous l'utilisez dans le script, vous devez disposer d'une instruction **LOAD** accompagnée d'une clause **Group by**.

Exemples :
Dans cet exemple, sont chargés tour à tour un fichier KML comportant des données de zones, puis une table comprenant les données de zones réduites et agrégées.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoReduceGeometry(world.Area,0.5)
as [ReducedArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

5.15 Fonctions d'interprétation

Les fonctions d'interprétation servent à interpréter le contenu de champs de texte d'entrée ou d'expressions, et imposent un format de données spécifié à la valeur numérique résultante. Ces fonctions vous permettent de définir le format du nombre, conformément au type de données, y compris les attributs tels que le séparateur décimal, le séparateur de milliers et le format de date.

Les fonctions d'interprétation renvoient toutes une valeur double comportant à la fois la chaîne et la valeur numérique, mais elles peuvent être considérées comme effectuant une conversion de chaîne en nombre. Les fonctions utilisent la valeur textuelle de l'expression d'entrée et génèrent un nombre représentant la chaîne.

En revanche, les fonctions de formatage ont un comportement inverse : elles prennent les expressions numériques et les évaluent en tant que de chaînes, en spécifiant le format d'affichage du texte résultant.

Si aucune fonction d'interprétation n'est utilisée, Qlik Sense interprète les données comme un mélange de nombres, de dates, d'heures, d'horodatages et de chaînes, en utilisant les paramètres par défaut définis par les variables de script et par le système d'exploitation pour les formats de nombre, de date et d'heure.

Les fonctions d'interprétation s'utilisent toutes aussi bien dans les scripts de chargement de données que dans les expressions de graphique.

Toutes les représentations numériques sont données avec un point comme séparateur décimal.

Vue d'ensemble des fonctions d'interprétation

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Date#

Date# évalue une expression comme une date dans le format spécifié dans le deuxième argument (si précisé). Si le code de format est omis, c'est le format de date par défaut défini dans le système d'exploitation qui est utilisé.

```
Date# (page 1267)(text[, format])
```

Interval#

Interval#() évalue une expression de texte comme un intervalle de temps dans le format défini sur le système d'exploitation, par défaut, ou dans le format spécifié dans le deuxième argument (si précisé).

```
Interval# (page 1268)(text[, format])
```

Money#

Money#() convertit une chaîne de texte en valeur monétaire en utilisant le format défini dans le script de chargement ou le système d'exploitation, à moins qu'une chaîne de format ne soit précisée. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

```
Money# (page 1269)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() interprète une chaîne de texte en une valeur numérique, c'est-à-dire qu'il convertit la chaîne d'entrée en un nombre au format spécifié dans le deuxième paramètre. En cas d'omission du deuxième paramètre, il utilise les séparateurs de décimaux et de milliers définis dans le script de chargement de données. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

```
Num# (page 1270)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() oblige l'expression à être traitée comme du texte, même si une interprétation numérique en est possible.

```
Text(expr)
```

Time#

Time#() évalue une expression en tant que valeur horaire en utilisant le format horaire défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée..

```
Time# (page 1271)(text[, format])
```

Timestamp#

Timestamp#() évalue une expression en tant que valeur de date et heure en utilisant le format d'horodatage défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

```
Timestamp# (page 1272)(text[, format])
```

Voir aussi :

p *Fonctions de formatage (page 1230)*

Date#

Date# évalue une expression comme une date dans le format spécifié dans le deuxième argument (si précisé).

Syntaxe :

```
Date#(text[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne décrivant le format de la chaîne de texte à évaluer. En cas d'omission, le format de date défini dans les variables système du script de chargement de données ou du système d'exploitation est utilisé.

Exemples et résultats :
L'exemple suivant utilise le format de date **M/D/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données.

Ajoutez cet exemple de script à votre application et exécutez-le.

```
Load *,
Num(Date#(StringDate)) as Date;
LOAD * INLINE [
StringDate
8/7/97
```

```
8/6/1997
]
```

Si vous créez une table utilisant **StringDate** et **Date** comme dimensions, les résultats sont comme suit :

Résultats

StringDate	Date
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() évalue une expression de texte comme un intervalle de temps dans le format défini sur le système d'exploitation, par défaut, ou dans le format spécifié dans le deuxième argument (si précisé).

Syntaxe :

```
Interval#(text[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne décrivant le format d'entrée attendu à utiliser lors de la conversion de la chaîne en intervalle numérique. En cas d'omission, ce sont le format de date abrégé, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés.

La fonction **interval#** convertit un intervalle de temps textuel en équivalent numérique.

Exemples et résultats :
Soient les paramètres de système d'exploitation suivants dans les exemples ci-dessous :

- Format de date abrégé : YY-MM-DD
- Format de l'heure : M/D/YY
- Séparateur décimal des nombres : .

Résultats

Exemple	Résultat
`Interval#( A, 'D hh:mm' )` où A='1 09:00'	1.375

Money#

Money#() convertit une chaîne de texte en valeur monétaire en utilisant le format défini dans le script de chargement ou le système d'exploitation, à moins qu'une chaîne de format ne soit précisée. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

Syntaxe :

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne décrivant le format d'entrée attendu à utiliser lors de la conversion de la chaîne en intervalle numérique. En cas d'omission, c'est le format monétaire défini dans le système d'exploitation qui est utilisé.
dec_sep	Chaîne indiquant le séparateur de nombres décimaux. En cas d'omission, la valeur MoneyDecimalSep définie dans le script de chargement de données est utilisée.
thou_sep	Chaîne indiquant le séparateur de milliers. En cas d'omission, la valeur MoneyThousandSep définie dans le script de chargement de données est utilisée.

La fonction **money#** se comporte généralement exactement comme la fonction **num#**, à ceci près qu'elle récupère ses valeurs par défaut pour le séparateur décimal et le séparateur de milliers à partir des variables de script du format monétaire ou des paramètres système de devise.

Exemples et résultats :
Soient les deux paramètres de système d'exploitation suivants dans les exemples ci-dessous :

- Paramètre par défaut de format monétaire 1 : kr # ##0,00
- Paramètre par défaut de format monétaire 2 : $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
où A=35 648,37 kr

Résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne	35 648.37 kr	35 648.37 kr
Nombre	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
où A= $35,648.37

Résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne	$35,648.37	$35,648.37
Nombre	35648.37	35648.37

Num#

Num#() interprète une chaîne de texte en une valeur numérique, c'est-à-dire qu'il convertit la chaîne d'entrée en un nombre au format spécifié dans le deuxième paramètre. En cas d'omission du deuxième paramètre, il utilise les séparateurs de décimaux et de milliers définis dans le script de chargement de données. Les symboles personnalisés de séparateur décimal et séparateur des milliers sont des paramètres facultatifs.

Syntaxe :

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Type de données renvoyé : double

La fonction **Num#()** renvoie une valeur double comportant à la fois la chaîne et la valeur numérique. La fonction prend la représentation textuelle de l'expression d'entrée et génère un nombre. Elle ne modifie pas le format du nombre : la sortie est formatée de la même manière que l'entrée.

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne spécifiant le format numérique utilisé dans le premier paramètre. En cas d'omission, les séparateurs de décimaux et de milliers définis dans le script de chargement de données sont utilisés.
dec_sep	Chaîne indiquant le séparateur de nombres décimaux. En cas d'omission, la valeur de la variable DecimalSep définie dans le script de chargement de données est utilisée.
thou_sep	Chaîne indiquant le séparateur de milliers. En cas d'omission, la valeur de la variable ThousandSep définie dans le script de chargement de données est utilisée.

Exemples et résultats :
La table suivante montre le résultat de *Num#(A, '#', '.', ',')* pour différentes valeurs de A.

Résultats

La présence d'une icône	Représentation de chaîne	Valeur numérique (affichée ici avec une virgule décimale)
35,648.31	35,648.31	35648.31
35 648.312	35 648.312	35648.312
35.648,3123	35.648,3123	-
35 648,31234	35 648,31234	-

Text

Text() oblige l'expression à être traitée comme du texte, même si une interprétation numérique en est possible.

Syntaxe :

```
Text (expr)
```

Type de données renvoyé : double

Exemple :

```
Text( A )
```

où A=1234

Résultats

Chaîne	Nombre
1234	-

Exemple :

```
Text( pi( ) )
```

Résultats

Chaîne	Nombre
3.1415926535898	-

Time#

Time#() évalue une expression en tant que valeur horaire en utilisant le format horaire défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée..

Syntaxe :

```
time#(text[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne décrivant le format de la chaîne de texte à évaluer. En cas d'omission, ce sont le format de date court, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés.

Exemple :

- Paramètre par défaut de format horaire 1 : hh:mm:ss
- Paramètre par défaut de format horaire 2 : hh.mm.ss

```
time#( A )
```
où A=09:00:00

Résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	09:00:00	09:00:00
Nombre :	0.375	-

Exemple :

- Paramètre par défaut de format horaire 1 : hh:mm:ss
- Paramètre par défaut de format horaire 2 : hh.mm.ss

```
time#( A, 'hh.mm' )
```
où A=09.00

Résultats

Résultats	Paramètre 1	Paramètre 2
Chaîne :	09.00	09.00
Nombre :	0.375	0.375

Timestamp#

Timestamp#() évalue une expression en tant que valeur de date et heure en utilisant le format d'horodatage défini dans les variables système du script de chargement de données ou sur le système d'exploitation, à moins qu'une chaîne de format ne soit précisée.

Syntaxe :

```
timestamp#(text[, format])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
text	Chaîne de texte à évaluer.
format	Chaîne décrivant le format de la chaîne de texte à évaluer. En cas d'omission, ce sont le format de date court, le format horaire et le séparateur décimal définis dans le système d'exploitation qui sont utilisés. ISO 8601 est pris en charge pour les horodatages.

Exemple :

L'exemple suivant utilise le format de date **M/D/YYYY**. Le format de date est indiqué dans l'instruction **SET DateFormat** située en haut de votre script de chargement de données.

Ajoutez cet exemple de script à votre application et exécutez-le.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
Chaîne
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Si vous créez une table utilisant **String** et **TS** comme dimensions, les résultats sont comme suit :

Résultats

Chaîne	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

5.16 Fonctions d'inter-enregistrements

Les fonctions d'inter-enregistrements s'utilisent dans les cas suivants :

- Dans le script de chargement de données, lorsque l'évaluation de l'enregistrement actif nécessite une valeur provenant d'enregistrements de données déjà chargés.
- Dans une expression de graphique, lorsqu'il est nécessaire d'utiliser une autre valeur de l'ensemble de données d'une visualisation.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation d'une fonction de graphique d'inter-enregistrement dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez une fonction de graphique d'inter-enregistrement dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via la fonction d'inter-enregistrement. Cette restriction ne s'applique pas à la fonction de script équivalente, le cas échéant.

Le référencement circulaire des définitions d'expression ne peut être réalisé de manière fiable que dans les tables comptant moins de 100 lignes, mais cela peut varier en fonction du matériel sur lequel le moteur Qlik est exécuté.

Fonctions de ligne

Ces fonctions s'utilisent uniquement dans les expressions de graphique.

Above

Above() évalue une expression au niveau de la ligne située au-dessus de la ligne active dans un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne située directement au-dessus. Pour les autres graphiques que les tables, l'évaluation de la fonction **Above()** porte sur la ligne située au-dessus de la ligne active dans l'équivalent du tableau simple du graphique.

```
Above - fonction de graphique([TOTAL [<fld{,fld}>]] expr [ , offset
[,count]])
```

Below

La fonction **Below()** évalue une expression au niveau de la ligne située en dessous de la ligne active dans un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne située directement en dessous. Pour les autres graphiques que les tables, l'évaluation de la fonction **Below()** porte sur la ligne située en dessous de la colonne active dans l'équivalent du tableau simple du graphique.

```
Below - fonction de graphique([TOTAL[<fld{,fld}>]] expression [ , offset
[,count ]])
```

Bottom

La fonction **Bottom()** évalue une expression au niveau de la dernière ligne (du bas) d'un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne du bas. Pour les autres graphiques que les tables, l'évaluation porte sur la dernière ligne de la colonne active dans l'équivalent du tableau simple du graphique.

```
Bottom - fonction de graphique([TOTAL[<fld{,fld}>]] expr [ , offset [,count
]])
```

Top

La fonction **Top()** évalue une expression au niveau de la première ligne (du haut) d'un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne du haut. Pour les autres graphiques que les tables, l'évaluation de la fonction **Top()** porte sur la première ligne de la colonne active dans l'équivalent du tableau simple du graphique.

```
Top - fonction de graphique([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() renvoie le nombre de lignes du segment de colonne actif d'un tableau. Pour les graphiques bitmap, **NoOfRows()** renvoie le nombre de lignes dans l'équivalent du tableau simple du graphique.

```
NoOfRows - fonction de graphique([TOTAL])
```

Fonctions de colonne

Ces fonctions s'utilisent uniquement dans les expressions de graphique.

Column

Column() renvoie la valeur détectée dans la colonne correspondant au numéro **ColumnNo** d'un tableau simple, quelles que soient les dimensions. Par exemple, **Column(2)** renvoie la valeur de la deuxième colonne de mesure.

```
Column - fonction de graphique(ColumnNo)
```

Dimensionality

Dimensionality() renvoie le nombre de dimensions correspondant à la ligne active. Dans le cas des tableaux croisés dynamiques, la fonction renvoie le nombre de colonnes de dimension présentant un contenu non agrégatif, c'est-à-dire ne comprenant pas de sommes partielles ou d'agrégats réduits.

```
Dimensionality - fonction de graphique ( )
```

Secondarydimensionality

SecondaryDimensionality() renvoie le nombre de lignes de dimension du tableau croisé dynamique qui ont du contenu non agrégatif, c'est-à-dire qui ne comprennent pas de sommes partielles ou d'agrégats réduits. Cette fonction est l'équivalent de la fonction **dimensionality()** pour les dimensions horizontales du tableau croisé dynamique.

```
SecondaryDimensionality - fonction de graphique ( )
```

Fonctions de champ

FieldIndex

FieldIndex() renvoie la position de la valeur de champ **value** du champ **field_name** (dans l'ordre de chargement).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() renvoie la valeur détectée à la position **elem_no** du champ **field_name** (dans l'ordre de chargement).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() est une fonction **entière** qui renvoie le nombre de valeurs distinctes dans un champ.

```
FieldValueCount (field_name)
```

Fonctions de tableau croisé dynamique

Ces fonctions s'utilisent uniquement dans les expressions de graphique.

After

After() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la colonne suivant la colonne active dans un segment de ligne du tableau.

```
After - fonction de graphique([TOTAL] expression [ , offset [,n]])
```

Before

Before() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la colonne précédant la colonne active dans un segment de ligne du tableau.

```
Before - fonction de graphique([TOTAL] expression [ , offset [,n]])
```

First

First() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la première colonne du segment de ligne actif du tableau croisé dynamique. Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

```
First - fonction de graphique([TOTAL] expression [ , offset [,n]])
```

Last

Last() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la dernière colonne du segment de ligne actif du tableau croisé dynamique. Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

```
Last - fonction de graphique([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() renvoie le numéro de la colonne active dans le segment de ligne actif d'un tableau croisé dynamique. La première colonne porte le nombre 1.

```
ColumnNo - fonction de graphique([TOTAL])
```

NoOfColumns

NoOfColumns() renvoie le nombre de colonnes dans le segment de ligne actif d'un tableau croisé dynamique.

```
NoOfColumns - fonction de graphique([TOTAL])
```

Fonctions d'inter-enregistrements utilisées dans le script de chargement de données

Exists

Exists() détermine si une valeur de champ donnée a déjà été chargée dans le champ du script de chargement de données. La fonction renvoie TRUE ou FALSE. Elle peut donc être utilisée dans la clause **where** d'une instruction **LOAD** ou d'une instruction **IF**.

```
Exists (field_name [, expr])
```

LookUp

Lookup() effectue des recherches dans une table déjà chargée et renvoie la valeur de **field_name** qui correspond à la première occurrence de la valeur **match_field_value** dans le champ **match_field_name**. La table peut désigner la table active ou une autre table chargée précédemment.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() renvoie la valeur d'un champ dans une table pour une ligne qui a déjà été chargée. Il est possible de spécifier le numéro de ligne et la table. Si aucune ligne n'est spécifiée, le dernier enregistrement précédemment chargé sera utilisé.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() recherche la valeur de l'expression **expr** en utilisant les données de l'enregistrement d'entrée précédent qui n'a pas été ignoré du fait d'une clause **where**. Dans le premier enregistrement d'une table interne, la fonction renvoie NULL.

```
Previous (page 1312)(expr)
```

Voir aussi :

p *Fonctions de plage (page 1333)*

Above - fonction de graphique

Above() évalue une expression au niveau de la ligne située au-dessus de la ligne active dans un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne située directement au-dessus. Pour les autres graphiques que les tables, l'évaluation de la fonction **Above()** porte sur la ligne située au-dessus de la ligne active dans l'équivalent du tableau simple du graphique.

Syntaxe :

```
Above([TOTAL] expr [ , offset [,count]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage offsetn supérieur à 0, l'évaluation de l'expression est déplacée de n lignes au-dessus de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction Above aboutit au même résultat que la fonction Below avec le décalage positif correspondant.
count	Si vous spécifiez un troisième argument **count** supérieur à 1, la fonction renvoie une plage de valeurs **count**, une pour chacune des lignes de table **count** situées au-dessus de la cellule de départ. De cette façon, la fonction peut être utilisée comme argument pour l'une des fonctions de plage spéciales. *Fonctions de plage (page 1333)*
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Pour la première ligne d'un segment de colonne, la fonction renvoie une valeur NULL, puisqu'il n'y a pas de ligne au-dessus.

Un segment de colonne se définit comme un sous-ensemble de cellules consécutives dotées des mêmes valeurs de dimensions dans l'ordre de tri actif. Les fonctions graphiques d'inter-enregistrements sont calculées dans le segment de colonne excluant la dimension située le plus à droite dans l'équivalent du tableau simple du graphique. Si le graphique ne comprend qu'une seule dimension ou si le qualificateur TOTAL est spécifié, l'évaluation de l'expression porte sur la table entière.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Limitations :

- Les appels récursifs renvoient la valeur NULL.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemples et résultats :

Example 1:

Visualisation de la table pour l'exemple 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

Dans la capture d'écran de la table présentée dans cet exemple, la visualisation de la table est créée à partir de la dimension **Customer** et des mesures `Sum(Sales)` et `Above(Sum(Sales))`.

La colonne Above(Sum(Sales)) renvoie NULL pour la ligne **Customer** contenant **Astrida**, puisqu'il n'y a aucune ligne au-dessus. Le résultat de la ligne **Betacab** affiche la valeur de Sum(Sales) pour **Astrida** tandis que le résultat relatif à **Canutility** indique la valeur de **Sum(Sales)** pour **Betacab** et ainsi de suite.

Pour la colonne intitulée Sum(Sales)+Above(Sum(Sales)), la ligne relative à **Betacab** affiche le résultat de l'addition des valeurs **Sum(Sales)** des lignes **Betacab** + **Astrida** (539+587). Le résultat de la ligne **Canutility** affiche le résultat de l'addition des valeurs **Sum(Sales)** des lignes **Canutility** + **Betacab** (683+539).

La mesure intitulée Above offset 3, créée à l'aide de l'expression `Sum(Sales)+Above(Sum(Sales), 3)`, comporte l'argument **offset**, défini sur 3, et a pour effet de prendre la valeur de la ligne située trois lignes au-dessus de la ligne active. Elle ajoute la valeur **Sum(Sales)** de la ligne **Customer** active à la valeur de la ligne **Customer** située trois lignes au-dessus. Les valeurs renvoyées pour les trois premières lignes **Customer** sont nulles.

La table indique également des mesures plus complexes : une valeur créée à partir de `Sum(Sales)+Above(Sum(Sales))` et une autre intitulée **Higher?**, créée à partir de `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Cette fonction peut également s'utiliser dans d'autres graphiques que les tables, dans les histogrammes par exemple.

Pour les autres types de graphique, convertissez le graphique en équivalent de tableau simple afin de pouvoir facilement interpréter la ligne à laquelle la fonction est liée.

Example 2:

Dans les captures d'écran des tables présentées dans cet exemple, d'autres dimensions ont été ajoutées aux visualisations : **Month** et **Product**. Pour les graphiques comportant plus d'une dimension, les résultats d'expressions contenant les fonctions **Above**, **Below**, **Top** et **Bottom** dépendent de l'ordre dans lequel les dimensions de colonne sont triées par Qlik Sense. Qlik Sense évalue les fonctions d'après les segments de colonne résultant de la dernière dimension qui a été triée. L'ordre de tri des colonnes est déterminé dans le panneau des propriétés, sous l'option **Tri**. Il ne correspond pas nécessairement à l'ordre d'affichage des colonnes dans une table.

Dans la capture d'écran suivante, qui présente la visualisation de la table de l'exemple 2, la dernière dimension triée étant **Month**, la fonction **Above** procède aux évaluations sur la base des mois. Une série de résultats est présentée pour chaque valeur **Product** associée à chaque mois (**Jan** à **Aug**) - un segment de colonne. Vient ensuite une série correspondant au segment de colonne suivant : chaque élément **Month** associé à l'élément **Product** suivant. Un segment de colonne est prévu pour chaque valeur **Customer** associée à chaque élément **Product**.

Visualisation de la table pour l'exemple 2

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	60
Astrida	AA	Apr	13	70
Astrida	AA	May	78	13
Astrida	AA	Jun	20	78
Astrida	AA	Jul	45	20
Astrida	AA	Aug	65	45

Example 3:

Dans la capture d'écran de la visualisation de la table de l'exemple 3, la dernière dimension triée correspond à **Product**. Pour obtenir ce résultat, déplacez la dimension Product en position 3 sous l'onglet Tri dans le panneau des propriétés. La fonction **Above** est évaluée pour chaque ligne **Product**. Étant donné qu'il n'y a que deux produits, **AA** et **BB**, chaque série ne comporte qu'un seul résultat différent de null. Sur la ligne **BB** du mois **Jan**, la valeur de **Above(Sum(Sales))** est 46. Pour la ligne **AA**, la valeur est nulle. La valeur de chaque ligne **AA** de chaque mois sera toujours nulle, car il n'existe pas de valeur de

Product au-dessus de AA. La seconde série est évaluée d'après **AA** et **BB** pour le mois **Feb**, pour la valeur **Customer**, **Astrida**. Dès lors que tous les mois ont été évalués pour **Astrida**, la séquence est répétée pour le second **Customer**Betacab, et ainsi de suite.

Visualisation de la table pour l'exemple 3

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	-
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	-
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	-
Astrida	BB	Apr	13	13

Exemple 4

Example 4:	**Résultat**
La fonction Above peut s'utiliser comme donnée d'entrée dans les fonctions de plage. Par exemple : `RangeAvg (Above(Sum (Sales),1,3))`.	Dans les arguments de la fonction Above(), offset est défini sur 1 et count est défini sur 3. La fonction recherche les résultats de l'expressionSum(Sales) dans les trois lignes situées immédiatement au-dessus de la ligne active dans le segment de colonne (lorsqu'il y a une ligne). Ces trois valeurs sont utilisées comme données d'entrée dans la fonction RangeAvg(), qui calcule la moyenne des valeurs de la plage de nombres fournie. Une table comprenant la dimension Customer donne les résultats suivants pour l'expression RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip : 603

Données utilisées dans les exemples :

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
```

```
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Below - fonction de graphique (page 1282)*
p *Bottom - fonction de graphique (page 1286)*
p *Top - fonction de graphique (page 1313)*
p *RangeAvg (page 1336)*

Below - fonction de graphique

La fonction **Below()** évalue une expression au niveau de la ligne située en dessous de la ligne active dans un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne située directement en dessous. Pour les autres graphiques que les tables, l'évaluation de la fonction **Below()** porte sur la ligne située en dessous de la colonne active dans l'équivalent du tableau simple du graphique.

Syntaxe :

```
Below([TOTAL] expr [ , offset [,count ]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
offset	Si vous spécifiez un décalage **offset**n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes en dessous de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction **Below** aboutit au même résultat que la fonction **Above** avec le décalage positif correspondant.
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de valeurs **count**, une pour chacune des lignes de table **count** situées en dessous de la cellule de départ. De cette façon, la fonction peut être utilisée comme argument pour l'une des fonctions de plage spéciales. *Fonctions de plage (page 1333)*
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Pour la dernière ligne d'un segment de colonne, la fonction renvoie une valeur NULL, puisqu'il n'y a pas de ligne en dessous.

Un segment de colonne se définit comme un sous-ensemble de cellules consécutives dotées des mêmes valeurs de dimensions dans l'ordre de tri actif. Les fonctions graphiques d'inter-enregistrements sont calculées dans le segment de colonne excluant la dimension située le plus à droite dans l'équivalent du tableau simple du graphique. Si le graphique ne comprend qu'une seule dimension ou si le qualificateur TOTAL est spécifié, l'évaluation de l'expression porte sur la table entière.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Limitations :

- Les appels récursifs renvoient la valeur NULL.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemples et résultats :

Example 1:

Visualisation de la table pour l'exemple 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	-	-	-	-

Dans la capture d'écran de l'exemple 1, la visualisation de la table présentée est créée à partir de la dimension **Customer** et des mesures `Sum(Sales)` et `Below(Sum(Sales))`.

La colonne **Below(Sum(Sales))** renvoie NULL pour la ligne **Customer** contenant **Divadip**, puisqu'il n'y a aucune ligne en dessous. Le résultat de la ligne **Canutility** affiche la valeur de Sum(Sales) pour **Divadip** tandis que le résultat relatif à **Betacab** indique la valeur de **Sum(Sales)** pour **Canutility** et ainsi de suite.

La table présente également des mesures plus complexes, affichées dans les colonnes étiquetées : `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** et **Higher?**. Ces expressions fonctionnent comme décrit dans les paragraphes suivants.

Pour la colonne intitulée **Sum(Sales)+Below(Sum(Sales))**, la ligne relative à **Astrida** affiche le résultat de l'addition des valeurs **Sum(Sales)** des lignes **Betacab** + **Astrida** (539+587). Le résultat de la ligne **Betacab** affiche le résultat de l'addition des valeurs **Sum(Sales)** de **Canutility** + **Betacab** (539+683).

La mesure intitulée **Below +Offset 3**, créée à l'aide de l'expression `Sum(Sales)+Below(Sum(Sales), 3)`, comporte l'argument **offset**, défini sur 3, et a pour effet de prendre la valeur de la ligne située trois lignes en dessous de la ligne active. Elle ajoute la valeur **Sum(Sales)** de la ligne **Customer** active à la valeur de la ligne **Customer** située trois lignes en dessous. Les valeurs renvoyées pour les trois dernières lignes **Customer** sont nulles.

La mesure étiquetée **Higher?** est créée à partir de l'expression :`IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Elle compare les valeurs de la ligne active dans la mesure **Sum(Sales)** à celles de la ligne située en dessous. Si la ligne active comporte une valeur plus élevée, le texte Higher est généré.

Cette fonction peut également s'utiliser dans d'autres graphiques que les tables, dans les histogrammes par exemple.

Pour les autres types de graphique, convertissez le graphique en équivalent de tableau simple afin de pouvoir facilement interpréter la ligne à laquelle la fonction est liée.

Pour les graphiques comportant plus d'une dimension, les résultats d'expressions contenant les fonctions **Above**, **Below**, **Top** et **Bottom** dépendent de l'ordre dans lequel les dimensions de colonne sont triées par Qlik Sense. Qlik Sense évalue les fonctions d'après les segments de colonne résultant de la dernière dimension qui a été triée. L'ordre de tri des colonnes est déterminé dans le panneau des propriétés, sous l'option **Tri**. Il ne correspond pas nécessairement à l'ordre d'affichage des colonnes dans une table.Pour plus de détails, reportez-vous à l'exemple : 2 dans la fonction **Above**.

Exemple 2

Example 2:	Résultat
La fonction **Below** peut s'utiliser comme donnée d'entrée dans les fonctions de plage. Par exemple : `RangeAvg (Below(Sum(Sales),1,3))`.	Dans les arguments de la fonction **Below()**, offset est défini sur 1 et count est défini sur 3. La fonction recherche les résultats de l'expression**Sum (Sales)** dans les trois lignes situées immédiatement en dessous de la ligne active dans le segment de colonne (lorsqu'il y a une ligne). Ces trois valeurs sont utilisées comme données d'entrée dans la fonction RangeAvg(), qui calcule la moyenne des valeurs de la plage de nombres fournie. Une table comprenant la dimension **Customer** donne les résultats suivants pour l'expression RangeAvg().
	Astrida 659.67 Betacab 720 Canutility 757 Divadip : -

Données utilisées dans les exemples :

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
```

```
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Above - fonction de graphique (page 1277)*
p *Bottom - fonction de graphique (page 1286)*
p *Top - fonction de graphique (page 1313)*
p *RangeAvg (page 1336)*

Bottom - fonction de graphique

La fonction **Bottom()** évalue une expression au niveau de la dernière ligne (du bas) d'un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne du bas. Pour les autres graphiques que les tables, l'évaluation porte sur la dernière ligne de la colonne active dans l'équivalent du tableau simple du graphique.

Syntaxe :

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage **offset**n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes au-dessus de la ligne inférieure. Si vous spécifiez un décalage négatif, la fonction **Bottom** aboutit au même résultat que la fonction **Top** avec le décalage positif correspondant.
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie non pas une valeur mais une plage de valeurs **count**, une pour chacune des **count** dernières lignes du segment de colonne actif. De cette façon, la fonction peut être utilisée comme argument pour l'une des fonctions de plage spéciales. *Fonctions de plage (page 1333)*
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Un segment de colonne se définit comme un sous-ensemble de cellules consécutives dotées des mêmes valeurs de dimensions dans l'ordre de tri actif. Les fonctions graphiques d'inter-enregistrements sont calculées dans le segment de colonne excluant la dimension située le plus à droite dans l'équivalent du tableau simple du graphique. Si le graphique ne comprend qu'une seule dimension ou si le qualificateur TOTAL est spécifié, l'évaluation de l'expression porte sur la table entière.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Limitations :

- Les appels récursifs renvoient la valeur NULL.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemples et résultats :

Visualisation de la table pour l'exemple 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

Dans la capture d'écran de la table présentée dans cet exemple, la visualisation de la table est créée à partir de la dimension **Customer** et des mesures `Sum(Sales)` et `Bottom(Sum(Sales))`.

La colonne **Bottom(Sum(Sales))** renvoie 757 pour toutes les lignes, car il s'agit de la valeur de la ligne inférieure : **Divadip**.

La table présente également des mesures plus complexes : une mesure créée à partir de `Sum(Sales)+Bottom(Sum(Sales))` et une autre intitulée **Bottom offset 3**, créée à l'aide de l'expression `Sum(Sales)+Bottom(Sum(Sales), 3)` et comportant l'argument **offset** défini sur 3. Elle ajoute la valeur **Sum(Sales)** de la ligne active à la valeur de la troisième ligne en partant de la ligne du bas, autrement dit, la ligne active plus la valeur correspondant à **Betacab**.

Exemple : 2

Dans les captures d'écran des tables présentées dans cet exemple, d'autres dimensions ont été ajoutées aux visualisations : **Month** et **Product**. Pour les graphiques comportant plus d'une dimension, les résultats d'expressions contenant les fonctions **Above**, **Below**, **Top** et **Bottom** dépendent de l'ordre dans lequel les dimensions de colonne sont triées par Qlik Sense. Qlik Sense évalue les fonctions d'après les segments de colonne résultant de la dernière dimension qui a été triée. L'ordre de tri des colonnes est déterminé dans le panneau des propriétés, sous l'option **Tri**. Il ne correspond pas nécessairement à l'ordre d'affichage des colonnes dans une table.

Dans la première table, l'expression est évaluée sur la base de la valeur **Month** tandis que dans la seconde table, elle est évaluée d'après la valeur **Product**. La mesure **End value** contient l'expression `Bottom(Sum (Sales))`. La ligne du bas pour **Month** correspond à Dec tandis que la valeur de Dec comporte les deux valeurs de **Product** indiquées dans la capture d'écran, soit 22. (Certaines lignes ont été masquées dans la capture d'écran pour économiser de l'espace.)

Première table pour l'exemple 2. Valeur de Bottom associée à la mesure End value d'après la ligne Month (Dec).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	22
Astrida	AA	Feb	60	22
Astrida	AA	Mar	70	22
Astrida	AA	Sep	78	22
Astrida	AA	Oct	12	22
Astrida	AA	Nov	78	22
Astrida	AA	Dec	22	22
Astrida	BB	Jan	46	22

Deuxième table pour l'exemple 2. Valeur de Bottom associée à la mesure End value d'après la ligne Product (BB pour Astrida).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Pour plus de détails, reportez-vous à l'exemple : 2 dans la fonction **Above**.

Exemple 3

Exemple : 3	**Résultat**
La fonction **Bottom** peut s'utiliser comme donnée d'entrée dans les fonctions de plage. Par exemple : `RangeAvg (Bottom(Sum(Sales),1,3))`.	Dans les arguments de la fonction **Bottom()**, offset est défini sur 1 et count est défini sur 3. La fonction recherche les résultats de l'expression**Sum (Sales)** dans les trois lignes, en commençant par la ligne située au-dessus de la ligne inférieure dans le segment de colonne (car offset=1), puis les deux lignes situées au-dessus de cette ligne (lorsqu'il y a une ligne). Ces trois valeurs sont utilisées comme données d'entrée dans la fonction RangeAvg(), qui calcule la moyenne des valeurs de la plage de nombres fournie. Une table comprenant la dimension **Customer** donne les résultats suivants pour l'expression RangeAvg().
	Astrida 659.67 Betacab 659.67 Canutility 659.67 Divadip : 659.67

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
```

```
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Top - fonction de graphique (page 1313)*

Column - fonction de graphique

Column() renvoie la valeur détectée dans la colonne correspondant au numéro **ColumnNo** d'un tableau simple, quelles que soient les dimensions. Par exemple, **Column(2)** renvoie la valeur de la deuxième colonne de mesure.

Syntaxe :

```
Column(ColumnNo)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
ColumnNo	Numéro de colonne d'une colonne de la table comportant une mesure. *La fonction Column() ne tient pas compte des colonnes de dimension.*

Limitations :

- Les appels récursifs renvoient la valeur NULL.
- Si **ColumnNo** fait référence à une colonne pour laquelle il n'existe aucune mesure, une valeur NULL est renvoyée.

- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemples et résultats :

Exemple : Pourcentage des ventes totales

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

Exemple : Pourcentage des ventes pour le client sélectionné

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

Exemples et résultats

Exemples	Résultats
Order Value est ajouté à la table en tant que mesure avec l'expression : `Sum (UnitPrice*UnitSales)`. Total Sales Value est ajouté en tant que mesure avec l'expression : `Sum (TOTAL UnitPrice*UnitSales)` % Sales est ajouté en tant que mesure avec l'expression `100*Column (1)/Column(2)`.	Le résultat de Column(1) provient de la colonne Order Value, car il s'agit de la première colonne de mesure. Le résultat de Column(2) provient de la colonne Total Sales Value, car il s'agit de la deuxième colonne de mesure. Observez les résultats obtenus dans la colonne % Sales de l'exemple *Pourcentage des ventes totales (page 1291)*.
Sélectionnez Customer A.	La sélection modifie la valeur Total Sales Value, a fortiori celle de %Sales. Consultez l'exemple *Pourcentage des ventes pour le client sélectionné (page 1291)*.

Données utilisées dans les exemples :

```
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 - fonction de graphique

Dimensionality() renvoie le nombre de dimensions correspondant à la ligne active. Dans le cas des tableaux croisés dynamiques, la fonction renvoie le nombre de colonnes de dimension présentant un contenu non agrégatif, c'est-à-dire ne comprenant pas de sommes partielles ou d'agrégats réduits.

Syntaxe :

```
Dimensionality ( )
```

Type de données renvoyé : entier

Limitations :

Cette fonction n'est disponible que pour les graphiques. Pour tous les types de graphique, excepté le tableau croisé dynamique, cette fonction renverra le nombre de dimensions dans toutes les lignes sauf celle du total, qui donnera 0.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemple : Expression de graphique utilisant Dimensionality

Exemple : Expression de graphique

Il est possible d'utiliser la fonction Dimensionality() avec un tableau croisé dynamique comme expression de graphique pour appliquer différents formatages de cellules suivant le nombre de dimensions d'une ligne contenant des données non agrégées. Cet exemple utilise la fonction Dimensionality() pour appliquer une couleur d'arrière-plan aux cellules du tableau correspondant à une condition donnée.

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple d'expression de graphique ci-dessous.

```
ProductSales:

Load * inline [
Country,Product,Sales,Budget
Sweden,AA,100000,50000
Germany,AA,125000,175000
Canada,AA,105000,98000
Norway,AA,74850,68500
Ireland,AA,49000,48000
Sweden,BB,98000,99000
Germany,BB,115000,175000
Norway,BB,71850,68500
Ireland,BB,31000,48000
] (delimiter is ',');
```

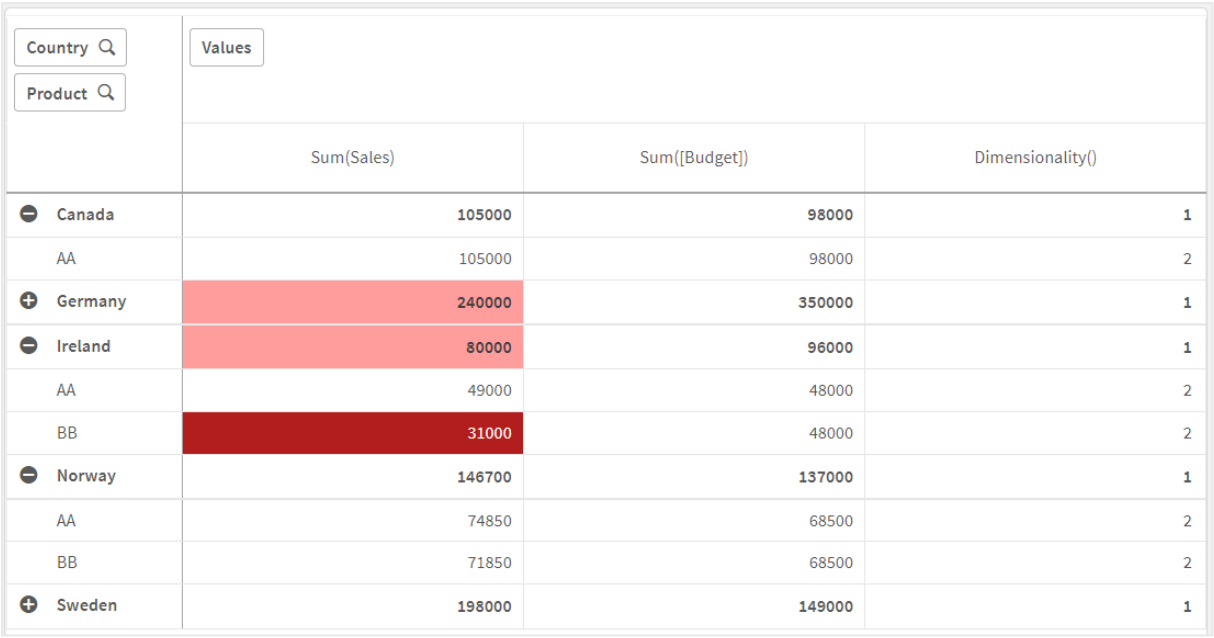
Expression de graphique

Créez une visualisation de tableau croisé dynamique dans une feuille Qlik Sense dotée des dimensions **Country** et **Product**. Ajoutez **Sum(Sales)**, **Sum(Budget)** et **Dimensionality()** comme mesures.

Dans le panneau des **Propriétés**, saisissez l'expression suivante comme **Expression de la couleur d'arrière-plan** pour la mesure **Sum(Sales)** :

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

Résultat :



Country / Product	Values		
	Sum(Sales)	Sum([Budget])	Dimensionality()
Canada	105000	98000	1
AA	105000	98000	2
Germany	240000	350000	1
Ireland	80000	96000	1
AA	49000	48000	2
BB	31000	48000	2
Norway	146700	137000	1
AA	74850	68500	2
BB	71850	68500	2
Sweden	198000	149000	1

Explication

L'expression `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If (Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` contient des instructions conditionnelles qui vérifient la valeur Dimensionality et les valeurs Sum(Sales) et Sum(Budget) pour chaque produit. Si les conditions sont remplies, une couleur d'arrière-plan est appliquée à la valeur Sum (Sales).

Exists

Exists() détermine si une valeur de champ donnée a déjà été chargée dans le champ du script de chargement de données. La fonction renvoie TRUE ou FALSE. Elle peut donc être utilisée dans la clause **where** d'une instruction **LOAD** ou d'une instruction **IF**.

> *Vous pouvez également déterminer si une valeur de champ n'a pas été chargée à l'aide de **Not Exists()**, mais nous vous conseillons d'être prudent en cas d'utilisation de la fonction **Not Exists()** dans une clause where. La fonction **Exists()** teste à la fois les tables précédemment chargées et les valeurs précédemment chargées dans la table active. Par conséquent, seule la première occurrence est chargée. Lorsque la deuxième occurrence est rencontrée, la valeur est déjà chargée. Pour plus d'informations, consultez les exemples.*

Syntaxe :

```
Exists(field_name [, expr])
```

Type de données renvoyé : booléen

Arguments :

Arguments

Argument	Description
field_name	Nom du champ dans lequel vous souhaitez rechercher une valeur. Vous pouvez utiliser un nom de champ explicite sans guillemets. Le champ doit déjà être chargé par le script. Autrement dit, vous ne pouvez pas faire référence à un champ qui est chargé dans une clause située plus loin dans le script.
expr	Valeur dont vous souhaitez vérifier l'existence. Vous pouvez utiliser une valeur explicite ou une expression qui fait référence à un ou plusieurs champs situés dans l'instruction de chargement. *Il n'est pas possible de faire référence à des champs non inclus dans l'instruction de chargement active.* Cet argument est facultatif. Si vous l'omettez, la fonction vérifiera si la valeur de **field_name** figurant dans l'enregistrement actif existe déjà.

Exemples et résultats :

Exemple 1

```
Exists (Employee)
```

Renvoie -1 (True) si la valeur du champ **Employee** figurant dans l'enregistrement actif existe déjà dans un enregistrement lu précédemment et contenant ce champ.

Les instructions `Exists (Employee, Employee)` et `Exists (Employee)` sont équivalentes.

Exemple 2

```
Exists(Employee, 'Bill')
```

Renvoie -1 (True) si la valeur de champ **'Bill'** se trouve dans le contenu actif du champ **Employee**.

Exemple 3

```
Employees: LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ]
(delimiter is '|');  Citizens: Load * inline [ Employee|Address Bill|New York Mary|London
Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where Exists (Employee);
Drop Tables Employees;
```

Vous obtenez une table que vous pouvez utiliser dans une visualisation de table à l'aide des dimensions Employee et Address.

La clause where, `where Exists (Employee)` signifie que seuls les noms issus de la table Citizens et figurant également dans Employees sont chargés dans la nouvelle table. Pour éviter toute confusion, l'instruction Drop supprime la table Employees.

Résultats

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Exemple 4

```
Employees: Load * inline [  Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Employee);   Drop Tables Employees;
```

La clause where inclut not : `where not Exists (Employee)`.

Autrement dit, seuls les noms issus de la table Citizens qui ne figurent pas dans Employees sont chargés dans la nouvelle table.

Notez la présence de deux valeurs pour Lucy dans la table Citizens, mais une seule figure dans la table des résultats. Lorsque vous chargez la première ligne avec la valeur Lucy, elle est incluse dans le champ Employee. De ce fait, lorsque la deuxième ligne est vérifiée, la valeur existe déjà.

Résultats

Employee	**Address**
Mary	London
Lucy	Madrid

Exemple 5

Cet exemple explique comment charger toutes les valeurs.

```
Employees: Load Employee As Name; LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Name, Employee);   Drop Tables Employees;
```

Pour obtenir toutes les valeurs associées à Lucy, il a fallu modifier deux éléments :

- Un chargement précédent dans la table Employees a été inséré, pour lequel le nom Employee a été remplacé par Name.
  ```
  Load Employee As Name;
  ```
- La condition Where de Citizens a été remplacée par :
  ```
  not Exists (Name, Employee).
  ```

Cela crée des champs pour Name et pour Employee. Lorsque la deuxième ligne avec Lucy est vérifiée, elle n'existe toujours pas dans Name.

Résultats

Employee	Address
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() renvoie la position de la valeur de champ **value** du champ **field_name** (dans l'ordre de chargement).

Syntaxe :

```
FieldIndex(field_name , value)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
field_name	Nom du champ pour lequel l'indice est requis. Par exemple, la colonne dans une table. À fournir sous forme de valeur de chaîne. Autrement dit, le nom du champ doit être placé entre guillemets simples.
value	Valeur du champ **field_name**.

Limitations :

- Si **value** est introuvable parmi les valeurs de champ du champ **field_name**, 0 est renvoyé.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction. Cette restriction ne s'applique pas à la fonction de script équivalente.

Exemples et résultats :

Les exemples suivants utilisent le champ : **First name** provenant de la table**Names**.

Exemples et résultats

Exemples	Résultats
Ajoutez l'exemple de données à votre application et exécutez cette dernière.	La table **Names** est chargée, comme dans les échantillons de données.
Fonction de graphique : dans une table contenant la dimension First name, ajoutez comme mesure :	
`FieldIndex ('First name','John')`	1, car 'John' apparaît en premier dans l'ordre de chargement du champ **First name**. En revanche, dans un volet de filtre, **John** figurerait en 2e position en commençant par le haut, car le contenu de la liste est trié par ordre alphabétique et pas selon l'ordre de chargement.
`FieldIndex ('First name','Peter')`	4, car **FieldIndex()** renvoie une seule valeur, la première occurrence dans l'ordre de chargement.
Fonction de script : étant donné que la table **Names** est chargée, comme dans les échantillons de données :	

`John1:`
`Load FieldIndex('First name','John') as MyJohnPos`
`Resident Names;`
	`MyJohnPos=`1, car 'John' apparaît en premier dans l'ordre de chargement du champ **First name**. En revanche, dans un volet de filtre, **John** figurerait en 2e position en commençant par le haut, car le contenu de la liste est trié par ordre alphabétique et pas selon l'ordre de chargement.

`Peter1:`
`Load FieldIndex('First name','Peter') as MyPeterPos`
`Resident Names;`
	`MyPeterPos=`4, car **FieldIndex()** renvoie une seule valeur, la première occurrence dans l'ordre de chargement.

Données utilisées dans l'exemple :

```
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

FieldValue() renvoie la valeur détectée à la position **elem_no** du champ **field_name** (dans l'ordre de chargement).

Syntaxe :

```
FieldValue(field_name , elem_no)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
field_name	Nom du champ pour lequel la valeur est requise. Par exemple, la colonne dans une table. À fournir sous forme de valeur de chaîne. Autrement dit, le nom du champ doit être placé entre guillemets simples.
elem_no	Numéro de position (élément) du champ, suivant l'ordre de chargement, pour lequel la valeur est renvoyée. Cela pourrait correspondre à la ligne dans une table, mais il dépend de l'ordre dans lequel les éléments (lignes) sont chargés.

Limitations :

- Si **elem_no** est supérieur au nombre de valeurs de champ, la chaîne NULL est renvoyée.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction. Cette restriction ne s'applique pas à la fonction de script équivalente.

Exemple

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple ci-dessous.

```
Names:

LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
```

```
Sue|Brown|SB|Yes
Mark|Carr|MC |No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:

Load FieldValue('First name',1) as MyPos1
Resident Names;

Peter1:

Load FieldValue('First name',5) as MyPos2
Resident Names;
```

Création d'une visualisation

Créez une visualisation de table dans une feuille Qlik Sense. Ajoutez les champs **First name**, **MyPos1** et **MyPos2** à la table.

Résultat

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Explication

Le résultat de **FieldValue('First name','1')** est John comme valeur pour **MyPos1** pour tous les prénoms, car John apparaît en premier dans l'ordre de chargement du champ **First name**. En revanche, dans un volet de filtre, John figurerait en 2e position en commençant par le haut, après Jane, car le contenu de la liste est trié par ordre alphabétique et pas selon l'ordre de chargement.

Le résultat de **FieldValue('First name','5')** est Jane comme valeur pour **MyPos2** pour tous les prénoms, car Jane apparaît en cinquième dans l'ordre de chargement du champ **First name**.

FieldValueCount

FieldValueCount() est une fonction **entière** qui renvoie le nombre de valeurs distinctes dans un champ.

Un chargement partiel peut supprimer des valeurs des données, qui ne seront pas reflétées dans le nombre renvoyé. Le nombre renvoyé correspondra à toutes les valeurs distinctes qui ont été chargées lors du chargement initial ou de tout chargement partiel ultérieur.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction. Cette restriction ne s'applique pas à la fonction de script équivalente.

Syntaxe :

```
FieldValueCount(field_name)
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
field_name	Nom du champ pour lequel la valeur est requise. Par exemple, la colonne dans une table. À fournir sous forme de valeur de chaîne. Autrement dit, le nom du champ doit être placé entre guillemets simples.

Exemples et résultats :

Les exemples suivants utilisent le champ **First name** provenant de la table **Names**.

Exemples et résultats

Exemples	Résultats
Ajoutez l'exemple de données à votre application et exécutez cette dernière.	La table **Names** est chargée, comme dans les échantillons de données.
Fonction de graphique : dans une table contenant la dimension First name, ajoutez comme mesure :	
`FieldValueCount('First name')`	5, car **Peter** apparaît deux fois.
`FieldValueCount('Initials')`	6, car **Initials** comprend uniquement des valeurs distinctes.
Fonction de script : étant donné que la table **Names** est chargée, comme dans les échantillons de données :	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1`=5, car 'Peter' apparaît deux fois.

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1`=6, car 'Initials' comprend uniquement des valeurs distinctes.

Données utilisées dans les exemples :

```
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

Lookup() effectue des recherches dans une table déjà chargée et renvoie la valeur de **field_name** qui correspond à la première occurrence de la valeur **match_field_value** dans le champ **match_field_name**. La table peut désigner la table active ou une autre table chargée précédemment.

Syntaxe :

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
field_name	Nom du champ pour lequel la valeur de renvoi est requise. La valeur saisie doit être une chaîne (par exemple, un littéral placé entre guillemets).
match_ field_name	Nom du champ dans lequel rechercher **match_field_value**. La valeur saisie doit être une chaîne (par exemple, un littéral placé entre guillemets).
match_ field_value	Valeur à rechercher dans le champ **match_field_name**.
table_name	Nom de la table dans laquelle rechercher la valeur. La valeur saisie doit être une chaîne (par exemple, un littéral placé entre guillemets). Si l'argument **table_name** est omis, la table active est alors renvoyée.

Les arguments sans guillemets renvoient à la table active. Pour faire référence à d'autres tables, placez un argument entre guillemets simples.

Limitations :

L'ordre de recherche correspond à l'ordre de chargement, sauf si la table est le résultat d'opérations complexes telles que des jointures, auquel cas l'ordre n'est pas bien défini. Les arguments **field_name** et **match_field_name** doivent désigner des champs faisant partie de la même table, elle-même spécifiée par l'argument **table_name**.

En l'absence de correspondance, la valeur NULL est renvoyée.

Exemple

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer l'exemple ci-dessous.

```
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;
```

Création d'une visualisation

Créez une visualisation de table dans une feuille Qlik Sense. Ajoutez les champs **ProductID**, **InvoiceID**, **CustomerID**, **Units** et **CategoryID** à la table.

Résultat

Table des résultats

ProductID	InvoiceID	CustomerID	Unités	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

Explication

Les échantillons de données utilisent la fonction **Lookup()** sous la forme suivante :

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

La table **ProductList** est chargée en premier.

La fonction **Lookup()** permet de créer la table **OrderData**. Elle spécifie le troisième argument comme **ProductID**. Il s'agit du champ pour lequel la valeur doit être recherchée dans le deuxième argument **'ProductID'** dans la liste **ProductList**, comme indiqué par les guillemets simples placés autour.

La fonction renvoie la valeur pour '**Category**' (dans la table **ProductList**), chargée comme **CategoryID**.

L'instruction **drop** supprime la table **ProductList** du modèle de données, car elle n'est pas nécessaire, ce qui aboutit à la table **OrderData**.

La fonction Lookup() étant flexible, elle peut accéder à toutes les tables chargées précédemment. Cependant, elle est lente par rapport à la fonction Applymap().

Voir aussi :

p *ApplyMap (page 1325)*

NoOfRows - fonction de graphique

NoOfRows() renvoie le nombre de lignes du segment de colonne actif d'un tableau. Pour les graphiques bitmap, **NoOfRows()** renvoie le nombre de lignes dans l'équivalent du tableau simple du graphique.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Syntaxe :

```
NoOfRows([TOTAL])
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Exemple : Expression de graphique avec NoOfRows

Exemple - expression de graphique

Script de chargement

Chargez les données suivantes sous forme de chargement inline dans l'éditeur de chargement de données pour créer les exemples d'expression de graphique ci-dessous.

```
        Temp:
LOAD * inline [
Region|SubRegion|RowNo()|NoOfRows()
Africa|Eastern
Africa|Western
Americas|Central
Americas|Northern
Asia|Eastern
Europe|Eastern
Europe|Northern
Europe|Western
Oceania|Australia
] (delimiter is '|');
```

Expression de graphique

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Region** et **SubRegion**. Ajoutez `RowNo( )`, `NoOfRows()` et `NoOfRows(Total)` comme mesures.

Résultat

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

Explication

Dans cet exemple, l'ordre de tri porte sur la première dimension, Region. C'est pourquoi chaque segment de colonne se compose d'un groupe de régions qui a la même valeur, par exemple, Africa.

La colonne **RowNo()** indique le nombre de lignes de chaque segment de colonne, par exemple, il existe deux lignes pour la région Africa. La numérotation des lignes reprend alors à 1 pour le segment de colonne suivant, c'est-à-dire Americas.

La colonne **NoOfRows()** compte le nombre de lignes de chaque segment de colonne, par exemple, Europe a trois lignes dans le segment de colonne.

La colonne **NoOfRows(Total)** ignore les dimensions, à cause de l'argument `TOTAL` pour `NoOfRows()`, et compte les lignes du tableau.

Si la table a été triée sur la deuxième dimension, SubRegion, les segments de colonne seront basés sur cette dimension, si bien que la numérotation des lignes changera pour chaque SubRegion.

Voir aussi :

p *RowNo - fonction de graphique (page 586)*

Peek

Peek() renvoie la valeur d'un champ dans une table pour une ligne qui a déjà été chargée. Il est possible de spécifier le numéro de ligne et la table. Si aucune ligne n'est spécifiée, le dernier enregistrement précédemment chargé sera utilisé.

La fonction peek() est le plus souvent utilisée pour rechercher les limites pertinentes d'une table précédemment chargée, à savoir, la première ou la dernière valeur d'un champ spécifique. Dans la plupart des cas, cette valeur est stockée dans une variable à des fins d'utilisation ultérieure, par exemple, comme condition dans une boucle do-while.

Syntaxe :

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
field_name	Nom du champ pour lequel la valeur de renvoi est requise.La valeur saisie doit être une chaîne (par exemple, un littéral placé entre guillemets).

Argument	Description
row_no	Ligne de la table indiquant le champ requis. Il peut s'agir d'une expression, mais le résultat doit correspondre à un entier. 0 renvoie au premier enregistrement, 1 au deuxième et ainsi de suite. Les nombres négatifs indiquent l'ordre des enregistrements à partir de la fin de la table. -1 renvoie ainsi au dernier enregistrement lu. Si aucun argument **row_no** n'est spécifié, -1 est utilisé.
table_name	Étiquette de table sans les deux-points finaux. Si aucun argument **table_name** n'est spécifié, la table active est utilisée. En cas d'utilisation à l'extérieur de l'instruction **LOAD** ou pour faire référence à une autre table, l'argument **table_name** doit être inclus.

Limitations :

Cette fonction peut uniquement renvoyer des valeurs d'enregistrements déjà chargés. Cela signifie que dans le premier enregistrement d'une table, un appel utilisant -1 comme row_no renverra NULL.

Exemples et résultats :

Exemple 1

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
EmployeeDates: Load * Inline [ EmployeeCode|StartDate|EndDate 101|02/11/2010|23/06/2012
102|01/11/2011|30/11/2013 103|02/01/2012| 104|02/01/2012|31/03/2012 105|01/04/2012|31/01/2013
106|02/11/2013| ] (delimiter is '|');       First_last_Employee: Load  EmployeeCode,  Peek
('EmployeeCode',0,'EmployeeDates') As FirstCode,  Peek('EmployeeCode',-1,'EmployeeDates') As
LastCode Resident EmployeeDates;
```

Tableau des résultats

Code d'employé	StartDate	EndDate	FirstCode	LastCode
101	02/11/2010	23/06/2012	101	106
102	01/11/2011	30/11/2013	101	106
103	02/01/2012		101	106
104	02/01/2012	31/03/2012	101	106
105	01/04/2012	31/01/2013	101	106
106	02/11/2013		101	106

FirstCode = 101, car `Peek('EmployeeCode',0, 'EmployeeDates')` renvoie la première valeur de EmployeeCode de la table EmployeeDates.

LastCode = 106, car `Peek('EmployeeCode',-1, 'EmployeeDates')` renvoie la dernière valeur de EmployeeCode de la table EmployeeDates.

La substitution de la valeur de l'argument **row_no** renvoie les valeurs des autres lignes de la table, comme suit :

`Peek('EmployeeCode',2, 'EmployeeDates')` renvoie la troisième valeur, 103, de la table comme FirstCode.

Notez cependant que si vous ne spécifiez pas la table comme troisième argument **table_name** dans ces exemples, la fonction fait référence à la table active (dans ce cas, la table interne).

Exemple 2

Si vous souhaitez accéder aux données qui se trouvent plus bas dans un tableau, vous devez le faire en deux étapes : tout d'abord, vous devez charger le tableau complet dans un tableau temporaire, puis le retrier à l'aide de **Peek()**.

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
T1: LOAD * inline [ ID|Value 1|3 1|4 1|6 3|7 3|8 2|1 2|11 5|2 5|78 5|13 ] (delimiter is '|');
T2:  LOAD *, IF(ID=Peek('ID'), Peek('List')&','&Value,Value) AS List RESIDENT T1 ORDER BY ID ASC; DROP TABLE T1;
```

Create a table in a sheet in your app with **ID**, **List**, and **Value** as the dimensions.

Tableau des résultats

ID	Liste	Valeur
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

L'instruction **IF()** est créée à partir de la table temporaire T1.
`Peek('ID')` référence le champ ID de la ligne précédente dans la table active T2.
`Peek('List')` référence le champ List de la ligne précédente dans la table T2, en cours de création pendant l'évaluation de l'expression.

L'instruction est évaluée de la manière suivante :
Si la valeur active d'ID est identique à la valeur précédente d'ID, alors indiquez la valeur de Peek('List') concaténée avec la valeur active de Value. Sinon, indiquez uniquement la valeur active de Value.

Si Peek('List') contient déjà un résultat concaténé, le nouveau résultat de Peek('List') sera concaténé avec lui.

*Notez la clause **Order by**. Elle spécifie le mode de tri de la table (par ID et selon un ordre croissant). Sans cela, la fonction Peek() utilise n'importe quel ordre arbitraire inclus dans la table interne, ce qui peut aboutir à des résultats imprévisibles.*

Exemple 3

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
Amounts: Load  Date#(Month,'YYYY-MM') as Month, Amount, Peek(Amount) as AmountMonthBefore
Inline [Month,Amount 2022-01,2 2022-02,3 2022-03,7 2022-04,9 2022-05,4 2022-06,1];
```

Tableau des résultats

Montant	AmountMonthBefore	Month
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

Le champ AmountMonthBefore contiendra le montant du mois précédent.

Ici, les paramètres row_no et table_name sont omis, c'est pourquoi les valeurs par défaut sont utilisées. Dans cet exemple, les trois appels de fonction suivants sont équivalents :

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

L'utilisation de -1 comme row_no signifie que la valeur de la ligne précédente sera utilisée. En substituant cette valeur, il est possible de récupérer les valeurs des autres lignes du tableau :

Peek(Amount,2) renvoie la troisième valeur du tableau : 7.

Exemple 4

Pour obtenir les résultats corrects, il convient de trier correctement les données, mais, malheureusement, cela n'est pas toujours le cas. De plus, la fonction Peek() ne peut pas être utilisée pour référencer des données qui n'ont pas encore été chargées. Si vous utilisez des tables temporaires et si vous passez plusieurs fois en revue les données, vous éviterez de tels problèmes.

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
tmp1Amounts: Load * Inline [Month,Product,Amount 2022-01,B,3 2022-01,A,8 2022-02,B,4 2022-02,A,6 2022-03,B,1 2022-03,A,6 2022-04,A,5 2022-04,B,5 2022-05,B,6 2022-05,A,7 2022-06,A,4 2022-06,B,8];  tmp2Amounts: Load *, If(Product=Peek(Product),Peek(Amount)) as AmountMonthBefore Resident tmp1Amounts Order By Product, Month Asc;  Drop Table tmp1Amounts; Amounts: Load *, If(Product=Peek(Product),Peek(Amount)) as AmountMonthAfter Resident tmp2Amounts Order By Product, Month Desc;  Drop Table tmp2Amounts;
```

Explication

Le tableau initial est trié en fonction du mois, ce qui signifie que la fonction peek(), dans de nombreux cas, renverra le montant pour le mauvais produit. C'est pourquoi il convient de retrier ce tableau. Pour ce faire, repassez en revue les données une deuxième fois en créant un nouveau tableau intitulé tmp2Amounts. Notez la clause Order by. Elle commence par trier les enregistrements par produit, puis par mois dans l'ordre croissant.

La fonction If() est nécessaire, car AmountMonthBefore doit être calculé uniquement si la ligne précédente contient les données du même produit, mais pour le mois précédent. Si le produit de la ligne en cours est comparé au produit de la ligne précédente, il est possible de valider cette condition.

Lors de la création du deuxième tableau, le premier tableau tmp1Amounts est abandonné via une instruction Drop Table.

Pour finir, les données sont passées en revue une troisième fois, mais, cette fois-ci, avec les mois triés dans l'ordre inverse. Cela permet de calculer également AmountMonthAfter.

Les clauses Order by spécifient le mode de tri du tableau ; sans elles, la fonction Peek() utilise n'importe quel ordre arbitraire inclus dans le tableau interne, ce qui peut aboutir à des résultats imprévisibles.

Résultat

Tableau des résultats

Month	Product	Montant	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5
2022-01	A	5	6	7

Month	Product	Montant	AmountMonthBefore	AmountMonthAfter
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8
2022-05	A	4	7	-
2022-06	B	8	6	-

Exemple 5

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
T1: Load * inline [ Quarter, Value 2003q1, 10000 2003q1, 25000 2003q1, 30000 2003q2, 1250 2003q2, 55000 2003q2, 76200 2003q3, 9240 2003q3, 33150 2003q3, 89450 2003q4, 1000 2003q4, 3000 2003q4, 5000 2004q1, 1000 2004q1, 1250 2004q1, 3000 2004q2, 5000 2004q2, 9240 2004q2, 10000 2004q3, 25000 2004q3, 30000 2004q3, 33150 2004q4, 55000 2004q4, 76200 2004q4, 89450 ];  T2: Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal; Load Quarter, sum(Value) as SumVal resident T1 group by Quarter;
```

Résultat

Tableau des résultats

Trimestre	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Explication

L'instruction LOAD **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** inclut un appel récursif dans lequel les valeurs précédentes sont ajoutées à la valeur actuelle. Cette opération est utilisée pour calculer une accumulation de valeurs dans le script.

Voir aussi :

Previous

Previous() recherche la valeur de l'expression **expr** en utilisant les données de l'enregistrement d'entrée précédent qui n'a pas été ignoré du fait d'une clause **where**. Dans le premier enregistrement d'une table interne, la fonction renvoie NULL.

Syntaxe :

```
Previous(expr)
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer. L'expression peut contenir des fonctions **previous()** imbriquées pour permettre l'accès à des enregistrements antérieurs. La fonction recherche les données directement dans la source d'entrée, ce qui vous permet de faire aussi référence à des champs qui n'ont pas été chargés dans Qlik Sense, c'est-à-dire même s'ils n'ont pas été stockés dans sa base de données associative.

Limitations :

Dans le premier enregistrement d'une table interne, la fonction renvoie NULL.

Exemple :

Saisissez les éléments suivants dans le script de chargement

```
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 '|');
```

En utilisant la fonction **Previous()** dans l'instruction **Load**, nous pouvons comparer la valeur active de Sales avec la valeur précédente et l'employer dans un troisième champ, Increase.

Table des résultats

Mois	Sales	Augmentation
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4
6	21	0
7	22	1
8	23	1
9	32	9
10	35	3
11	40	5
12	41	1

Top - fonction de graphique

La fonction **Top()** évalue une expression au niveau de la première ligne (du haut) d'un segment de colonne d'une table. La ligne pour laquelle elle est calculée dépend de la valeur de décalage **offset** (si présente), le paramètre par défaut étant la ligne du haut. Pour les autres graphiques que les tables, l'évaluation de la fonction **Top()** porte sur la première ligne de la colonne active dans l'équivalent du tableau simple du graphique.

Syntaxe :

```
Top([TOTAL] expr [ , offset [,count ]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage **offset** de n, supérieur à 1, l'évaluation de l'expression est déplacée de n lignes en dessous de la ligne supérieure. Si vous spécifiez un décalage négatif, la fonction **Top** aboutit au même résultat que la fonction **Bottom** avec le décalage positif correspondant.

Argument	Description
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de **count** valeurs, une pour chacune des **count** dernières lignes du segment de colonne actif. De cette façon, la fonction peut être utilisée comme argument pour l'une des fonctions de plage spéciales. *Fonctions de plage (page 1333)*
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Un segment de colonne se définit comme un sous-ensemble de cellules consécutives dotées des mêmes valeurs de dimensions dans l'ordre de tri actif. Les fonctions graphiques d'inter-enregistrements sont calculées dans le segment de colonne excluant la dimension située le plus à droite dans l'équivalent du tableau simple du graphique. Si le graphique ne comprend qu'une seule dimension ou si le qualificateur TOTAL est spécifié, l'évaluation de l'expression porte sur la table entière.

Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Limitations :

- Les appels récursifs renvoient la valeur NULL.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemples et résultats :

Exemple : 1

Dans la capture d'écran de la table présentée dans cet exemple, la visualisation de la table est créée à partir de la dimension **Customer** et des mesures `Sum(Sales)` et `Top(Sum(Sales))`.

La colonne **Top(Sum(Sales))** renvoie 587 pour toutes les lignes, car il s'agit de la valeur de la ligne supérieure : **Astrida**.

La table présente également des mesures plus complexes : une mesure créée à partir de `Sum(Sales)+Top(Sum(Sales))` et une autre intitulée **Top offset 3**, créée à l'aide de l'expression `Sum(Sales)+Top(Sum`

(Sales), 3) et comportant l'argument **offset** défini sur 3. Elle ajoute la valeur **Sum(Sales)** de la ligne active à la valeur de la troisième ligne en partant de la ligne du haut, autrement dit, la ligne active plus la valeur correspondant à **Canutility**.

Exemple 1

Top and Bottom

Customer	Sum(Sales)	Top(Sum(Sales))	Sum(Sales)+Top(Sum(Sales))	Top offset 3
Totals	2566	587	3153	3249
Astrida	587	587	1174	1270
Betacab	539	587	1126	1222
Canutility	683	587	1270	1366
Divadip	757	587	1344	1440

Exemple : 2

Dans les captures d'écran des tables présentées dans cet exemple, d'autres dimensions ont été ajoutées aux visualisations : **Month** et **Product**. Pour les graphiques comportant plus d'une dimension, les résultats d'expressions contenant les fonctions **Above**, **Below**, **Top** et **Bottom** dépendent de l'ordre dans lequel les dimensions de colonne sont triées par Qlik Sense. Qlik Sense évalue les fonctions d'après les segments de colonne résultant de la dernière dimension qui a été triée. L'ordre de tri des colonnes est déterminé dans le panneau des propriétés, sous l'option **Tri**. Il ne correspond pas nécessairement à l'ordre d'affichage des colonnes dans une table.

Première table pour l'exemple 2. Valeur de Top associée à la mesure First value d'après la ligne Month (Jan).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	46
Astrida	AA	Apr	13	46
Astrida	AA	May	78	46
Astrida	AA	Jun	20	46
Astrida	AA	Jul	45	46
Astrida	AA	Aug	65	46
Astrida	AA	Sep	78	46
Astrida	AA	Oct	12	46
Astrida	AA	Nov	78	46
Astrida	AA	Dec	22	46

Deuxième table pour l'exemple 2. Valeur de Top associée à la mesure First value d'après la ligne Product (AA pour Astrida).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Pour plus de détails, reportez-vous à l'exemple : 2 dans la fonction **Above**.

Exemple 3

Exemple : 3	**Résultat**
La fonction **Top** peut s'utiliser comme donnée d'entrée dans les fonctions de plage. Par exemple : `RangeAvg (Top(Sum(Sales),1,3))`.	Dans les arguments de la fonction **Top()**, offset est défini sur 1 et count est défini sur 3. La fonction recherche les résultats de l'expression **Sum (Sales)** dans les trois lignes, en commençant par la ligne située en dessous de la ligne inférieure dans le segment de colonne (car offset=1) et les deux lignes situées en dessous (lorsqu'il y a une ligne). Ces trois valeurs sont utilisées comme données d'entrée dans la fonction RangeAvg(), qui calcule la moyenne des valeurs de la plage de nombres fournie. Une table comprenant la dimension **Customer** donne les résultats suivants pour l'expression RangeAvg().
	Astrida 603 Betacab 603 Canutility 603 Divadip : 603

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
```

```
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Voir aussi :

p *Bottom - fonction de graphique (page 1286)*
p *Above - fonction de graphique (page 1277)*
p *Sum - fonction de graphique (page 349)*
p *RangeAvg (page 1336)*
p *Fonctions de plage (page 1333)*

SecondaryDimensionality - fonction de graphique

SecondaryDimensionality() renvoie le nombre de lignes de dimension du tableau croisé dynamique qui ont du contenu non agrégatif, c'est-à-dire qui ne comprennent pas de sommes partielles ou d'agrégats réduits. Cette fonction est l'équivalent de la fonction **dimensionality()** pour les dimensions horizontales du tableau croisé dynamique.

Syntaxe :

```
SecondaryDimensionality( )
```

Type de données renvoyé : entier

Limitations :

- À moins d'être utilisée dans des tableaux croisés dynamiques, la fonction **SecondaryDimensionality** renvoie toujours 0.
- Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou une table, le tri de la visualisation revient à l'entrée triée via cette fonction.

After - fonction de graphique

After() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la colonne suivant la colonne active dans un segment de ligne du tableau.

Syntaxe :

```
after([TOTAL] expr [, offset [, count ]])
```

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

Arguments :

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage **offset** n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes à droite de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction **After** aboutit au même résultat que la fonction **Before** avec le décalage positif correspondant.
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de valeurs, une pour chacune des lignes de la table jusqu'à la valeur **count**, en comptant vers la droite par rapport à la cellule de départ.
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Pour la dernière colonne d'un segment de ligne, la fonction renvoie une valeur NULL, puisqu'il n'y a pas de colonne après.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs. L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Exemple :

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` renvoie une moyenne des trois résultats de la fonction **sum(x)** évaluée dans les trois colonnes situées immédiatement à droite de la colonne active.

Before - fonction de graphique

Before() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la colonne précédant la colonne active dans un segment de ligne du tableau.

Syntaxe :

```
before([TOTAL] expr [, offset [, count]])
```

Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage **offset** n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes à gauche de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction **Before** aboutit au même résultat que la fonction **After** avec le décalage positif correspondant.
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de valeurs, une pour chacune des lignes de la table jusqu'à la valeur **count**, en comptant vers la gauche par rapport à la cellule de départ.
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Pour la première colonne d'un segment de ligne, la fonction renvoie une valeur NULL, puisqu'il n'y a pas de colonne avant.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs.L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Exemples :

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` renvoie une moyenne des trois résultats de la fonction **sum(x)** évaluée dans les trois colonnes immédiatement à gauche de la colonne active.

First - fonction de graphique

First() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la première colonne du segment de ligne actif du tableau croisé dynamique. Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Syntaxe :

```
first([TOTAL] expr [, offset [, count]])
```

Arguments :

Arguments

Argument	Description
expression	Expression ou champ contenant les données à mesurer.
offset	Si vous spécifiez un décalage **offset** n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes à droite de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction **First** aboutit au même résultat que la fonction **Last** avec le décalage positif correspondant.

Argument	Description
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de valeurs, une pour chacune des lignes de la table jusqu'à la valeur **count**, en comptant vers la droite par rapport à la cellule de départ.
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs.L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Exemples :

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` renvoie une moyenne des résultats de la fonction **sum(x)** évaluée sur les cinq colonnes situées le plus à gauche par rapport au segment de ligne actif.

Last - fonction de graphique

Last() renvoie la valeur d'une expression évaluée avec les valeurs de dimension d'un tableau croisé dynamique telles qu'elles figurent dans la dernière colonne du segment de ligne actif du tableau croisé dynamique. Cette fonction renvoie NULL dans tous les types de graphique autres que les tableaux croisés dynamiques.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Syntaxe :

```
last([TOTAL] expr [, offset [, count]])
```

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.

Argument	Description
offset	Si vous spécifiez un décalage **offset** n supérieur à 1, l'évaluation de l'expression est déplacée de n lignes à gauche de la ligne active. Si vous spécifiez un décalage égal à 0, l'expression est évaluée sur la ligne active. Si vous spécifiez un décalage négatif, la fonction **First** aboutit au même résultat que la fonction **Last** avec le décalage positif correspondant.
count	Si vous spécifiez un troisième paramètre **count** supérieur à 1, la fonction renvoie une plage de valeurs, une pour chacune des lignes de la table jusqu'à la valeur **count**, en comptant vers la gauche par rapport à la cellule de départ.
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs.L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Exemple :

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` renvoie une moyenne des résultats de la fonction **sum(x)** évaluée sur les cinq colonnes situées le plus à droite par rapport au segment de ligne actif.

ColumnNo - fonction de graphique

ColumnNo() renvoie le numéro de la colonne active dans le segment de ligne actif d'un tableau croisé dynamique. La première colonne porte le nombre 1.

Syntaxe :

```
ColumnNo([total])
```

Arguments :

Arguments

Argument	Description
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs.L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Exemple :

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - fonction de graphique

NoOfColumns() renvoie le nombre de colonnes dans le segment de ligne actif d'un tableau croisé dynamique.

Le tri sur les valeurs des ordonnées dans les graphiques ou le tri par colonnes d'expressions dans les tableaux n'est pas autorisé lors de l'utilisation de cette fonction de graphique dans l'une des expressions du graphique. Ces options de tri sont donc automatiquement désactivées. Lorsque vous utilisez cette fonction de graphique dans une visualisation ou un tableau, le tri de la visualisation revient à l'entrée triée via cette fonction.

Syntaxe :

```
NoOfColumns([total])
```

Arguments :

Arguments

Argument	Description
TOTAL	Si la table est unidimensionnelle ou si le qualificateur **TOTAL** est utilisé comme argument, le segment de colonne actif est toujours égal à la colonne entière.

Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimensions, à l'exception de la ligne affichant la dernière dimension dans l'ordre de tri inter-champs.L'ordre de tri inter-champs pour les dimensions horizontales des tableaux croisés dynamiques est simplement défini par l'ordre des dimensions de haut en bas..

Exemple :

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Fonctions logiques

Cette section décrit les fonctions de gestion des opérations logiques. Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

IsNum

Renvoie -1 (True) si l'expression peut être interprétée comme un nombre ; sinon, renvoie 0 (False).

```
IsNum( expr )
```

IsText

Renvoie -1 (True) si l'expression a une représentation textuelle ; sinon, renvoie 0 (False).

```
IsText( expr )
```

*Les fonctions **IsNum** et **IsText** renvoient toutes deux 0 si l'expression correspond à NULL.*

Exemple :

L'exemple suivant charge une table intégrée comportant à la fois des valeurs textuelles et des valeurs numériques, et ajoute deux champs destinés à vérifier le type de chaque valeur (numérique ou textuelle).

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

La table résultante a l'aspect suivant :

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Fonctions de mappage

Cette section décrit les fonctions de gestion des tables de mappage. Les tables de mappage permettent de remplacer des valeurs ou des noms de champ lors de l'exécution du script.

Les fonctions de mappage s'emploient exclusivement dans le script de chargement de données.

Vue d'ensemble des fonctions de mappage

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

ApplyMap

La fonction de script **ApplyMap** permet de mapper le résultat d'une expression à une table de mappage déjà chargée.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

La fonction de script **MapSubstring** permet de mapper des parties d'une expression à une table de mappage déjà chargée. Le mappage est sensible à la casse des caractères et n'est pas itératif ; les sous-chaînes sont mappées de gauche à droite.

```
MapSubstring ('mapname', expr)
```

ApplyMap

La fonction de script **ApplyMap** permet de mapper le résultat d'une expression à une table de mappage déjà chargée.

Syntaxe :

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
map_name	Nom d'une table de mappage créée précédemment à l'aide de l'instruction **mapping load** ou **mapping select**. Son nom doit être mis entre guillemets simples droits. *Si vous utilisez cette fonction dans une variable étendue par macro et faites référence à une table de mappage qui n'existe pas, l'appel de fonction se solde par un échec et aucun champ n'est créé.*
expression	Expression dont le résultat doit être mappé.
default_ mapping	Si l'argument est indiqué, cette valeur est utilisée comme valeur par défaut lorsque la table de mappage ne contient pas de valeur correspondante pour expression. Si l'argument n'est pas précisé, la valeur de l'expression est renvoyée telle quelle.

Le nom du champ de sortie d'ApplyMap doit être différent de celui des champs d'entrée. Cela pourrait entraîner des résultats inattendus. Exemple à ne pas suivre : `ApplyMap('Map', A) as A`.

Exemple :

Dans cet exemple, nous chargeons une liste de représentants commerciaux accompagnés du code pays représentant leur pays de résidence. Nous utilisons une table mappant un code pays à un pays pour remplacer le code pays par le nom du pays. Seulement trois pays sont définis dans la table de mappage ; les autres codes pays sont mappés à l'entrée 'Rest of the world' (Autre pays).

```
// 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';
```

La table résultante (Salespersons) a l'aspect suivant :

Resulting table

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

MapSubstring

La fonction de script **MapSubstring** permet de mapper des parties d'une expression à une table de mappage déjà chargée. Le mappage est sensible à la casse des caractères et n'est pas itératif ; les sous-chaînes sont mappées de gauche à droite.

Syntaxe :

```
MapSubstring('map_name', expression)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
map_name	Nom d'une table de mappage lue précédemment par une instruction **mapping load** ou **mapping select**. Ce nom doit être mis entre guillemets simples droits. *Si vous utilisez cette fonction dans une variable étendue par macro et faites référence à une table de mappage qui n'existe pas, l'appel de fonction se solde par un échec et aucun champ n'est créé.*
expression	Expression dont le résultat doit être mappé par des sous-chaînes.

Exemple :

Dans cet exemple, nous chargeons une liste de modèles de produit. Chaque modèle possède un ensemble d'attributs décrits par un code composé. En utilisant la table de mappage avec MapSubstring, nous pouvons étendre les codes d'attribut à une description.

```
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';
```

La table résultante a l'aspect suivant :

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

5.19 Fonctions mathématiques

Cette section décrit les fonctions des constantes mathématiques et des valeurs booléennes. Ces fonctions sont dépourvues de paramètres, mais les parenthèses sont tout de même requises.

Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

e

La fonction renvoie la base des logarithmes népériens, **e**.(2.71828...).

```
e( )
```

false

La fonction renvoie une valeur double avec la valeur textuelle 'False' et la valeur numérique 0, qui peuvent être utilisées comme un faux logique dans les expressions.

```
false( )
```

pi

La fonction renvoie la valeur de π (3.14159...).

```
pi( )
```

rand

La fonction renvoie un nombre aléatoire compris entre 0 et 1. Elle peut servir à créer des échantillons de données.

```
rand( )
```

Exemple :

Cet exemple de script crée une table de 1 000 enregistrements contenant des caractères en majuscules sélectionnés de manière aléatoire, c'est-à-dire des caractères compris dans la plage 65 à 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true

La fonction renvoie une valeur double avec la valeur textuelle 'True' et la valeur numérique -1, qui peuvent être utilisées comme un vrai logique dans les expressions.

```
true( )
```

5.20 Fonctions NULL

Cette section décrit les fonctions permettant de renvoyer ou de détecter des valeurs NULL.

Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

Vue d'ensemble des fonctions NULL

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

EmptyIsNull

La fonction **EmptyIsNull** convertit des chaînes vides en NULL. Ainsi, elle renvoie NULL si le paramètre est une chaîne vide ; sinon, elle renvoie le paramètre.

```
EmptyIsNull (expr )
```

IsNull

La fonction **IsNull** teste si la valeur d'une expression est NULL et, si tel est le cas, renvoie -1 (True). Dans le cas contraire, la fonction renvoie 0 (False).

```
IsNull (expr )
```

Null

La fonction **Null** renvoie la valeur NULL.

```
NULL( )
```

EmptyIsNull

La fonction **EmptyIsNull** convertit des chaînes vides en NULL. Ainsi, elle renvoie NULL si le paramètre est une chaîne vide ; sinon, elle renvoie le paramètre.

Syntaxe :

```
EmptyIsNull(exp )
```

Exemples et résultats :

Exemples de script

Exemple	Résultat
`EmptyIsNull(AdditionalComments)`	Cette expression renvoie comme nulle toutes les valeurs de chaîne vides du champ *AdditionalComments* au lieu de chaînes vides. Les chaînes non vides et les nombres sont renvoyés.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Cette expression supprime tous les tirets, espaces et parenthèses du champ *PhoneNumber*. S'il ne reste plus de caractères, la fonction EmptyIsNull renvoie la chaîne vide comme nulle ; un numéro de téléphone vide est identique à l'absence de numéro de téléphone.

IsNull

La fonction **IsNull** teste si la valeur d'une expression est NULL et, si tel est le cas, renvoie -1 (True). Dans le cas contraire, la fonction renvoie 0 (False).

Syntaxe :

```
IsNull(expr )
```

*Une chaîne d'une longueur égale à zéro n'est pas considérée comme NULL et entraîne la fonction **IsNull** à renvoyer la valeur False.*

Exemple : Script de chargement de données

Dans cet exemple, une table intégrée comportant quatre lignes est chargée, avec les trois premières lignes ne contenant rien, - ou 'NULL' dans la colonne Value. Nous convertissons ces valeurs en représentations de valeurs NULL réelles, avec l'instruction **LOAD** antérieure au milieu, à l'aide de la fonction **Null**.

La première instruction **LOAD** antérieure ajoute un champ destiné à vérifier si la valeur correspond à NULL, en utilisant la fonction **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];
```

Voici la table résultante. Dans la colonne ValueNullConv, les valeurs NULL sont représentées par -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

La fonction **Null** renvoie la valeur NULL.

Syntaxe :

```
Null( )
```

Exemple : Script de chargement de données

Dans cet exemple, une table intégrée comportant quatre lignes est chargée, avec les trois premières lignes ne contenant rien, - ou 'NULL' dans la colonne Value. Nous souhaitons convertir ces valeurs en représentations de valeurs NULL réelles.

L'instruction **LOAD** antérieure du milieu procède à la conversion en utilisant la fonction **Null**.

La première instruction **LOAD** antérieure ajoute un champ destiné à vérifier si la valeur est égale à NULL, à simple titre d'illustration dans cet exemple.

```
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];
```

Voici la table résultante. Dans la colonne ValueNullConv, les valeurs NULL sont représentées par -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Fonctions de plage

Les fonctions de plage sont des fonctions qui, à partir d'un tableau de valeurs, génèrent une valeur unique comme résultat. Les fonctions de plage s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

Par exemple, dans une visualisation, une fonction de plage peut calculer une valeur unique à partir d'un tableau d'inter-enregistrements. Dans le script de chargement de données, une fonction de plage peut calculer une valeur unique à partir d'un tableau de valeurs d'une table interne.

*Les fonctions de plage remplacent les fonctions numériques générales suivantes : **numsum**, **numavg**, **numcount**, **nummin** et **nummax**, qui doivent désormais être considérées comme obsolètes.*

Fonctions de plage de base

RangeMax

RangeMax() renvoie les valeurs numériques les plus élevées contenues dans l'expression ou le champ.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() renvoie la dernière valeur contenue dans l'expression ou le champ selon l'ordre de tri du texte.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() renvoie les valeurs numériques les plus basses contenues dans l'expression ou le champ.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() renvoie la première valeur contenue dans l'expression ou le champ selon l'ordre de tri du texte.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() permet de déterminer la valeur la plus fréquente (valeur de mode) contenue dans l'expression ou le champ.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() est une fonction double qui renvoie une valeur si l'expression a pour résultat une valeur unique. Dans le cas contraire, la valeur **NULL** est renvoyée.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() renvoie la somme d'une plage de valeurs. Toutes les valeurs non numériques sont traitées comme des 0.

```
RangeSum (first_expr[, Expression])
```

Fonctions de plage de décompte

RangeCount

RangeCount() renvoie le nombre de valeurs, à la fois textuelles et numériques, contenues dans l'expression ou le champ.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() renvoie le nombre de valeurs non numériques (NULL comprises) contenues dans l'expression ou le champ.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() permet de déterminer le nombre de valeurs NULL contenues dans l'expression ou le champ.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() permet de déterminer le nombre de valeurs numériques contenues dans une expression ou un champ.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() renvoie le nombre de valeurs textuelles contenues dans une expression ou un champ.

```
RangeTextCount (first_expr[, Expression])
```

Fonctions de plage statistiques

RangeAvg

RangeAvg() renvoie la moyenne d'une plage. Vous pouvez saisir une plage de valeurs ou une expression pour cette fonction.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() renvoie le coefficient de corrélation pour deux ensembles de données. Le coefficient de corrélation mesure la relation entre deux ensembles de données.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() renvoie la valeur correspondant au énième **fractile** (quantile) d'une plage de nombres.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() renvoie la valeur correspondant au coefficient d'aplatissement d'une plage de nombres.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() renvoie la valeur correspondant à l'asymétrie d'une plage de nombres.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() permet de déterminer l'écart type d'une plage de données.

```
RangeStdev (expr1[, Expression])
```

Fonctions de plage financières

RangeIRR

RangeIRR() renvoie le taux de rendement interne pour une série de flux de liquidités représentés par les valeurs d'entrée.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() renvoie la valeur actuelle nette d'un investissement sur la base d'un taux d'escompte et d'une série de paiements périodiques (valeurs négatives) et de revenus (valeurs positives) ultérieurs. Le résultat suit le format de nombre par défaut de **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() renvoie le taux de rendement interne pour un calendrier de flux de liquidités qui n'est pas nécessairement périodique. Pour calculer le taux de rendement interne pour une série de flux de liquidités périodiques, utilisez la fonction **RangeIRR**.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() renvoie la valeur actuelle nette pour un calendrier de flux de liquidités qui n'est pas nécessairement périodique. Le résultat utilise un format monétaire (de nombre) par défaut. Pour calculer la valeur actuelle nette pour une série de flux de liquidités périodiques, utilisez la fonction **RangeNPV**.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Voir aussi :

p *Fonctions d'inter-enregistrements (page 1273)*

RangeAvg

RangeAvg() renvoie la moyenne d'une plage. Vous pouvez saisir une plage de valeurs ou une expression pour cette fonction.

Syntaxe :

```
RangeAvg(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de script

Exemples	Résultats
`RangeAvg (1,2,4)`	Renvoie 2.33333333.
`RangeAvg (1,'xyz')`	Renvoie 1.
`RangeAvg (null( ), 'abc')`	Renvoie NULL.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeAvg pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Exemple contenant une expression :

```
RangeAvg (Above(MyField),0,3))
```

Renvoie une moyenne mobile du résultat de la plage de trois valeurs de **MyField** calculée sur la ligne active et les deux lignes au-dessus. En spécifiant 3 pour le troisième argument, la fonction **Above()** renvoie trois valeurs, s'il y a suffisamment de lignes au-dessus, utilisées comme données d'entrée dans la fonction **RangeAvg()**.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Puisqu'il s'agit de la ligne du haut, la plage ne comporte qu'une seule valeur.
2	6	Il n'y a qu'une seule ligne au-dessus de cette ligne, la plage est donc : 10,2.
8	6.6666666667	Équivalent de RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Voir aussi :

p *Avg - fonction de graphique (page 399)*
p *Count - fonction de graphique (page 354)*

RangeCorrel

RangeCorrel() renvoie le coefficient de corrélation pour deux ensembles de données. Le coefficient de corrélation mesure la relation entre deux ensembles de données.

Syntaxe :

```
RangeCorrel(x_value , y_value[, Expression])
```

Type de données renvoyé : numérique

Les séries de données doivent être saisies sous forme de paires (x,y). Par exemple, pour évaluer deux séries de données array 1 et array 2, où array 1 = 2,6,9 et array 2 = 3,8,4, vous devez spécifier `RangeCorrel (2,3,6,8,9,4)`, qui renvoie 0.269.

Arguments :

Arguments

Argument	Description
x-value, y-value	Chaque valeur représente une valeur unique ou une plage de valeurs renvoyées par une fonction d'inter-enregistrements avec un troisième paramètre facultatif. Chaque valeur ou plage de valeurs doit correspondre à une valeur **x-value** ou à une plage de valeurs **y-values**.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Cette fonction nécessite au moins deux paires de coordonnées pour être calculée.

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes renvoient NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeCorrel (2,3,6,8,9,4,8,5)`	Renvoie 0.2492. Il est possible de charger cette fonction dans le script ou de l'ajouter dans une visualisation via l'éditeur d'expression.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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 '|');
```

Dans une table comportant ID1 en tant que dimension et la mesure RangeCorrel (x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)), la fonction **RangeCorrel()** recherche la valeur de **Correl** dans la plage de six paires x,y, pour chacune des valeurs ID1.

Table des résultats

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Exemple :

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

Dans une table comportant RangeID en tant que dimension et la mesure RangeCorrel(Below (X,0,4,BelowY,0,4)), la fonction **RangeCorrel()** utilise les résultats des fonctions **Below()**, qui, du fait du troisième argument (count) défini sur 4, génèrent une plage de quatre valeurs x-y à partir de la table XY chargée.

Table des résultats

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

La valeur de RangeID 01 équivaut à saisir manuellement RangeCorrel(2,3,6,8,9,4,8,5). Pour les autres valeurs de RangeID, les séries générées par la fonction Below() sont les suivantes : (6,8,9,4,8,5), (9,4,8,5) et (8,5), cette dernière produisant un résultat nul.

Voir aussi :

p *Correl - fonction de graphique (page 403)*

RangeCount

RangeCount() renvoie le nombre de valeurs, à la fois textuelles et numériques, contenues dans l'expression ou le champ.

Syntaxe :

```
RangeCount(first_expr[, Expression])
```

Type de données renvoyé : entier

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à compter.
Expression	Expressions ou champs facultatifs contenant la plage de données à compter.

Limitations :

Les valeurs NULL ne sont pas comptées.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeCount (1,2,4)`	Renvoie 3.
`RangeCount (2,'xyz')`	Renvoie 2.
`RangeCount (null( ))`	Renvoie 0.
`RangeCount (2,'xyz', null())`	Renvoie 2.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeCount pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Exemple contenant une expression :

```
RangeCount (Above(MyField,1,3))
```

Renvoie le nombre de valeurs contenues dans les trois résultats de **MyField**. En spécifiant le premier argument de la fonction **Above()** comme 1 et le deuxième argument comme 3, elle renvoie les valeurs des trois premiers champs situés au-dessus de la ligne active, s'il y a suffisamment de lignes, utilisées comme données d'entrée dans la fonction **RangeCount()**.

Données utilisées dans les exemples :

Échantillons de données

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Voir aussi :

p *Count - fonction de graphique (page 354)*

RangeFractile

RangeFractile() renvoie la valeur correspondant au énième **fractile** (quantile) d'une plage de nombres.

RangeFractile() utilise une interpolation linéaire entre les classements les plus proches lors du calcul du fractile.

Syntaxe :

```
RangeFractile(fractile, first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
fractile	Nombre compris entre 0 et 1 correspondant au fractile (quantile exprimé sous forme de fraction) à calculer.
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeFractile (0.24,1,2,4,6)`	Renvoie 1.72.
`RangeFractile(0.5,1,2,3,4,6)`	Renvoie 3.
`RangeFractile (0.5,1,2,5,6)`	Renvoie 3.5.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeFrac pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Exemple contenant une expression :

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

Dans cet exemple, la fonction d'inter-enregistrements **Above()** contient les arguments offset et count facultatifs. Vous obtenez une plage de résultats pouvant être utilisés comme données d'entrée dans les fonctions de plage. Dans le cas présent, `Above(Sum(MyField),0,3)` renvoie les valeurs de `MyField` pour la ligne active et les deux lignes au-dessus. Ces valeurs servent de données d'entrée dans la fonction **RangeFractile()**. Donc, pour la ligne du bas de la table ci-dessous, il s'agit de l'équivalent de la fonction `RangeFractile(0.5, 3,4,6)`, c'est-à-dire le calcul du fractile 0.5 pour les séries 3, 4 et 6. Dans les deux premières lignes de la table ci-dessous, le nombre de valeurs de la plage est réduit en conséquence, lorsqu'il n'y a pas de lignes au-dessus de la ligne active. Des résultats similaires sont obtenus pour les autres fonctions d'inter-enregistrements.

Échantillons de données

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

Voir aussi :

p *Above - fonction de graphique (page 1277)*
p *Fractile - fonction de graphique (page 407)*

RangeIRR

RangeIRR() renvoie le taux de rendement interne pour une série de flux de liquidités représentés par les valeurs d'entrée.

Le taux de rendement interne correspond au taux d'intérêt perçu pour un investissement consistant en des paiements (valeurs négatives) et des revenus (valeurs positives) qui interviennent à intervalle régulier.

Syntaxe :

```
RangeIRR(value[, value][, Expression])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Valeur unique ou plage de valeurs renvoyée par une fonction d'inter-enregistrements avec un troisième paramètre facultatif. La fonction nécessite au moins une valeur positive et une valeur négative à calculer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemple de table

Exemples	Résultats
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Renvoie 0.0866.

Exemples	Résultats

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.
`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 '|');`
	La table résultante affiche les valeurs renvoyées par la fonction RangeIRR pour chaque enregistrement de la table.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

Voir aussi :

p *Fonctions d'inter-enregistrements (page 1273)*

RangeKurtosis

RangeKurtosis() renvoie la valeur correspondant au coefficient d'aplatissement d'une plage de nombres.

Syntaxe :

```
RangeKurtosis(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeKurtosis (1,2,4,7)`	Renvoie -0.28571428571429.

Voir aussi :

p *Kurtosis - fonction de graphique (page 415)*

RangeMax

RangeMax() renvoie les valeurs numériques les plus élevées contenues dans l'expression ou le champ.

Syntaxe :

```
RangeMax(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMax (1,2,4)`	Renvoie 4.
`RangeMax (1,'xyz')`	Renvoie 1.
`RangeMax (null( ), 'abc')`	Renvoie NULL.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeMax pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Exemple contenant une expression :

```
RangeMax (Above(MyField,0,3))
```

Renvoie la valeur maximale de la plage de trois valeurs de **MyField** calculée sur la ligne active et les deux lignes au-dessus. En spécifiant 3 pour le troisième argument, la fonction **Above()** renvoie trois valeurs, s'il y a suffisamment de lignes au-dessus, utilisées comme données d'entrée dans la fonction **RangeMax()**.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() renvoie la dernière valeur contenue dans l'expression ou le champ selon l'ordre de tri du texte.

Syntaxe :

```
RangeMaxString(first_expr[, Expression])
```

Type de données renvoyé : chaîne

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMaxString (1,2,4)`	Renvoie 4.
`RangeMaxString ('xyz','abc')`	Renvoie 'xyz'.
`RangeMaxString (5,'abc')`	Renvoie 'abc'.
`RangeMaxString (null( ))`	Renvoie NULL.

Exemple contenant une expression :

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Renvoie le dernier (dans l'ordre de tri du texte) des trois résultats de la fonction **MaxString(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Voir aussi :

p *MaxString - fonction de graphique (page 537)*

RangeMin

RangeMin() renvoie les valeurs numériques les plus basses contenues dans l'expression ou le champ.

Syntaxe :

```
RangeMin(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMin (1,2,4)`	Renvoie 1.
`RangeMin (1,'xyz')`	Renvoie 1.
`RangeMin (null( ), 'abc')`	Renvoie NULL.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeMin pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeMin
1	5
2	2

RangeID	MyRangeMin
3	2
4	9
5	5
6	2

Exemple contenant une expression :

```
RangeMin (Above(MyField,0,3)
```

Renvoie la valeur minimale de la plage de trois valeurs de **MyField** calculée sur la ligne active et les deux lignes au-dessus. En spécifiant 3 pour le troisième argument, la fonction **Above()** renvoie trois valeurs, s'il y a suffisamment de lignes au-dessus, utilisées comme données d'entrée dans la fonction **RangeMin()**.

Données utilisées dans les exemples :

Échantillons de données

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Voir aussi :

p *Min - fonction de graphique (page 340)*

RangeMinString

RangeMinString() renvoie la première valeur contenue dans l'expression ou le champ selon l'ordre de tri du texte.

Syntaxe :

```
RangeMinString(first_expr[, Expression])
```

Type de données renvoyé : chaîne

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMinString (1,2,4)`	Renvoie 1.
`RangeMinString ('xyz','abc')`	Renvoie 'abc'.
`RangeMinString (5,'abc')`	Renvoie 5.
`RangeMinString (null( ))`	Renvoie NULL.

Exemple contenant une expression :

```
RangeMinString (Above(MinString(MyField),0,3))
```

Renvoie le premier (dans l'ordre de tri du texte) des trois résultats de la fonction **MinString(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Voir aussi :

p *MinString - fonction de graphique (page 540)*

RangeMissingCount

RangeMissingCount() renvoie le nombre de valeurs non numériques (NULL comprises) contenues dans l'expression ou le champ.

Syntaxe :

```
RangeMissingCount(first_expr[, Expression])
```

Type de données renvoyé : entier

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à compter.
Expression	Expressions ou champs facultatifs contenant la plage de données à compter.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMissingCount (1,2,4)`	Renvoie 0.
`RangeMissingCount (5,'abc')`	Renvoie 1.
`RangeMissingCount (null( ))`	Renvoie 1.

Exemple contenant une expression :

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Renvoie le nombre de valeurs non numériques dans les trois résultats de la fonction **MinString(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Renvoie 2, car il n'y a pas de ligne au-dessus de celle-ci. 2 des 3 valeurs sont donc manquantes.
abc	2	Renvoie 2, car il n'y a qu'une seule ligne au-dessus de celle-ci et la ligne active est non numérique ('abc').
8	1	Renvoie 1, car 1 des 3 lignes comprend une valeur non numérique ('abc').
def	2	Renvoie 2, car 2 des 3 lignes comprennent des valeurs non numériques ('def' et 'abc').
xyz	2	Renvoie 2, car 2 des 3 lignes comprennent des valeurs non numériques (' xyz' et 'def').
9	2	Renvoie 2, car 2 des 3 lignes comprennent des valeurs non numériques (' xyz' et 'def').

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
```

```
8
'def'
'xyz'
9
] ;
```

Voir aussi :

p *MissingCount - fonction de graphique (page 358)*

RangeMode

RangeMode() permet de déterminer la valeur la plus fréquente (valeur de mode) contenue dans l'expression ou le champ.

Syntaxe :

```
RangeMode(first_expr {, Expression})
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si plusieurs valeurs présentent la fréquence la plus élevée, la valeur NULL est renvoyée.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeMode (1,2,9,2,4)`	Renvoie 2.
`RangeMode ('a',4,'a',4)`	Renvoie NULL.
`RangeMode (null( ))`	Renvoie NULL.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction **MyRangeMode** pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

Exemple contenant une expression :

```
RangeMode (Above(MyField,0,3))
```

Renvoie la valeur la plus fréquente dans les trois résultats de **MyField** évaluée sur la ligne active et les deux lignes au-dessus. En spécifiant 3 pour le troisième argument, la fonction **Above()** renvoie trois valeurs, s'il y a suffisamment de lignes au-dessus, utilisées comme données d'entrée dans la fonction **RangeMode()**.

Données utilisées dans l'exemple :

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeMode(Above(MyField,0,3))
10	Renvoie 10, car il n'y a aucune ligne au-dessus. La valeur unique est donc celle qui est la plus fréquente.
2	-
8	-
18	-
5	-
9	-

Voir aussi :

p *Mode - fonction de graphique (page 343)*

RangeNPV

RangeNPV() renvoie la valeur actuelle nette d'un investissement sur la base d'un taux d'escompte et d'une série de paiements périodiques (valeurs négatives) et de revenus (valeurs positives) ultérieurs. Le résultat suit le format de nombre par défaut de **money**.

Pour les flux de liquidités qui ne sont pas nécessairement périodiques, voir *RangeXNPV (page 1370)*.

Syntaxe :

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
discount_rate	Taux d'intérêt par période.
value	Paiement ou revenu intervenant au terme de chaque période. Chaque valeur peut représenter une valeur unique ou une plage de valeurs renvoyée par une fonction d'inter-enregistrements avec un troisième paramètre facultatif.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Exemples	Résultats
`RangeNPV(0.1,-10000,3000,4200,6800)`	Renvoie 1188.44.

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.
`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 '|');`
	La table résultante affiche les valeurs renvoyées par la fonction RangeNPV pour chaque enregistrement de la table.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

Voir aussi :

p *Fonctions d'inter-enregistrements (page 1273)*

RangeNullCount

RangeNullCount() permet de déterminer le nombre de valeurs NULL contenues dans l'expression ou le champ.

Syntaxe :

```
RangeNullCount(first_expr [, Expression])
```

Type de données renvoyé : entier

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeNullCount (1,2,4)`	Renvoie 0.
`RangeNullCount (5,'abc')`	Renvoie 0.
`RangeNullCount (null( ), null( ))`	Renvoie 2.

Exemple contenant une expression :

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Renvoie le nombre de valeurs NULL dans les trois résultats de la fonction **Sum(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

*Si vous copiez **MyField** dans l'exemple ci-dessous, la valeur NULL ne sera pas renvoyée.*

Échantillons de données

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Renvoie 2, car il n'y a pas de ligne au-dessus de celle-ci. 2 des 3 valeurs sont donc manquantes (=NULL).
'abc'	Renvoie 1, car il n'y a qu'une seule ligne au-dessus de celle-ci. Une des trois valeurs est donc manquante (=NULL).
8	Renvoie 0, car aucune des trois lignes ne correspond à une valeur NULL.

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Voir aussi :

p *NullCount - fonction de graphique (page 361)*

RangeNumericCount

RangeNumericCount() permet de déterminer le nombre de valeurs numériques contenues dans une expression ou un champ.

Syntaxe :

```
RangeNumericCount(first_expr[, Expression])
```

Type de données renvoyé : entier

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeNumericCount (1,2,4)`	Renvoie 3.
`RangeNumericCount (5,'abc')`	Renvoie 1.
`RangeNumericCount (null( ))`	Renvoie 0.

Exemple contenant une expression :

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Renvoie le nombre de valeurs numériques dans les trois résultats de la fonction **MaxString(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Voir aussi :

p *NumericCount - fonction de graphique (page 363)*

RangeOnly

RangeOnly() est une fonction double qui renvoie une valeur si l'expression a pour résultat une valeur unique. Dans le cas contraire, la valeur **NULL** est renvoyée.

Syntaxe :

```
RangeOnly(first_expr[, Expression])
```

Type de données renvoyé : double

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples	Résultats
`RangeOnly (1,2,4)`	Renvoie NULL.
`RangeOnly (5,'abc')`	Renvoie NULL.
`RangeOnly (null( ), 'abc')`	Renvoie 'abc'.
`RangeOnly(10,10,10)`	Renvoie 10.

Voir aussi :

p *Only - fonction de graphique (page 346)*

RangeSkew

RangeSkew() renvoie la valeur correspondant à l'asymétrie d'une plage de nombres.

Syntaxe :

```
RangeSkew(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`rangeskew (1,2,4)`	Renvoie 0.93521952958283.
`rangeskew (above (SalesValue,0,3))`	Renvoie une asymétrie mobile de la plage des trois valeurs renvoyées par la fonction above() calculée sur la ligne active et les deux lignes au-dessus.

Données utilisées dans l'exemple :

Échantillons de données

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
] ;
```

Voir aussi :

p *Skew - fonction de graphique (page 447)*

RangeStdev

RangeStdev() permet de déterminer l'écart type d'une plage de données.

Syntaxe :

```
RangeStdev(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

Si la fonction ne trouve aucune valeur numérique, elle renvoie la valeur NULL.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeStdev (1,2,4)`	Renvoie 1.5275252316519.
`RangeStdev (null( ))`	Renvoie NULL.
`RangeStdev (above (SalesValue),0,3))`	Renvoie une valeur type mobile de la plage des trois valeurs renvoyées par la fonction above() calculée sur la ligne active et les deux lignes au-dessus.

Données utilisées dans l'exemple :

Échantillons de données

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
] ;
```

Voir aussi :

p *Stdev - fonction de graphique (page 450)*

RangeSum

RangeSum() renvoie la somme d'une plage de valeurs. Toutes les valeurs non numériques sont traitées comme des 0.

Syntaxe :

```
RangeSum(first_expr[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Arguments

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Limitations :

La fonction **RangeSum** traite toutes les valeurs non numériques comme des 0.

Exemples et résultats :

Exemples

Exemples	Résultats
`RangeSum (1,2,4)`	Renvoie 7.
`RangeSum (5,'abc')`	Renvoie 5.
`RangeSum (null( ))`	Renvoie 0.

Exemple :

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.

```
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
];
```

La table résultante affiche les valeurs renvoyées par la fonction MyRangeSum pour chaque enregistrement de la table.

Table des résultats

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Exemple contenant une expression :

```
RangeSum (Above(MyField,0,3))
```

Renvoie la somme des trois valeurs de **MyField)** : à partir de la ligne active et des deux lignes au-dessus. En spécifiant 3 pour le troisième argument, la fonction **Above()** renvoie trois valeurs, s'il y a suffisamment de lignes au-dessus, utilisées comme données d'entrée dans la fonction **RangeSum()**.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Échantillons de données

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Voir aussi :

p *Sum - fonction de graphique (page 349)*
p *Above - fonction de graphique (page 1277)*

RangeTextCount

RangeTextCount() renvoie le nombre de valeurs textuelles contenues dans une expression ou un champ.

Syntaxe :

```
RangeTextCount(first_expr[, Expression])
```

Type de données renvoyé : entier

Arguments :

Les arguments de cette fonction peuvent contenir des fonctions d'inter-enregistrements, qui renvoient à leur tour une liste de valeurs.

Argument

Argument	Description
first_expr	Expression ou champ contenant les données à mesurer.
Expression	Expressions ou champs facultatifs contenant la plage de données à mesurer.

Exemples et résultats :

Exemples de fonction

Exemples	Résultats
`RangeTextCount (1,2,4)`	Renvoie 0.
`RangeTextCount (5,'abc')`	Renvoie 1.
`RangeTextCount (null( ))`	Renvoie 0.

Exemple contenant une expression :

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Renvoie le nombre de valeurs textuelles dans les trois résultats de la fonction **MaxString(MyField)** évaluée sur la ligne active et les deux lignes au-dessus.

Données utilisées dans les exemples :

*Désactivez la fonction de tri de **MyField** pour vous assurer que l'exemple fonctionne comme prévu.*

Données d'exemple

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

Données utilisées dans les exemples :

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Voir aussi :

p *TextCount - fonction de graphique (page 367)*

RangeXIRR

RangeXIRR() renvoie le taux de rendement interne pour un calendrier de flux de liquidités qui n'est pas nécessairement périodique. Pour calculer le taux de rendement interne pour une série de flux de liquidités périodiques, utilisez la fonction **RangeIRR**.

Syntaxe :

```
RangeXIRR(value, date{, value, date})
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
value	Flux de liquidités ou série de flux de liquidités qui correspond à un calendrier de paiements à dates. La série de valeurs doit contenir au moins une valeur positive et une valeur négative.
date	Date de paiement ou calendrier de dates de paiement qui correspond aux paiements de flux de liquidités.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Tous les paiements sont actualisés sur une base de 365 jours par an.

Exemples	Résultats
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Renvoie 0.1532.

Voir aussi :

p *RangeIRR (page 1345)*

RangeXNPV

RangeXNPV() renvoie la valeur actuelle nette pour un calendrier de flux de liquidités qui n'est pas nécessairement périodique. Le résultat utilise un format monétaire (de nombre) par défaut. Pour calculer la valeur actuelle nette pour une série de flux de liquidités périodiques, utilisez la fonction **RangeNPV**.

Syntaxe :

```
RangeXNPV(discount_rate, values, dates[, Expression])
```

Type de données renvoyé : numérique

Arguments :

Arguments

Argument	Description
discount_ rate	Taux d'intérêt par période.

Argument	Description
values	Flux de liquidités ou série de flux de liquidités qui correspond à un calendrier de paiements à dates. Chaque valeur peut représenter une valeur unique ou une plage de valeurs renvoyée par une fonction d'inter-enregistrements avec un troisième paramètre facultatif. La série de valeurs doit contenir au moins une valeur positive et une valeur négative.
dates	Date de paiement ou calendrier de dates de paiement qui correspond aux paiements de flux de liquidités.

Limitations :

Les valeurs textuelles, les valeurs NULL et les valeurs manquantes sont ignorées.

Tous les paiements sont actualisés sur une base de 365 jours par an.

Exemple de table

Exemples	Résultats
`RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')`	Renvoie 80.25.

Ajoutez l'exemple de script à votre application et exécutez-le. Pour afficher le résultat, ajoutez les champs répertoriés dans la colonne de résultats à une feuille de votre application.
`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 '|');`
	La table résultante affiche les valeurs renvoyées par la fonction RangeXNPV pour chaque enregistrement de la table.
	RangeID — RangeXNPV
	1 — $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

5.22 Fonctions de classement et de clustering

Ces fonctions s'utilisent uniquement dans les expressions de graphique.

Fonctions de classement dans les graphiques

La suppression des valeurs zéro est automatiquement désactivée lorsque ces fonctions sont utilisées. Les valeurs NULL sont ignorées.

Rank

Rank() évalue les lignes du graphique dans l'expression et, pour chaque ligne, affiche la position relative de la valeur de la dimension évaluée dans l'expression. Lors de l'évaluation de l'expression, la fonction compare le résultat à celui des autres lignes contenant le segment de colonne actif et renvoie le classement de la ligne active dans ce segment.

```
Rank - fonction de graphique([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() évalue l'expression et compare le résultat à celui des autres colonnes contenant le segment de ligne actif d'un tableau croisé dynamique. La fonction renvoie ensuite le classement de la colonne active dans le segment.

```
HRank - fonction de graphique([TOTAL] expr[, mode[, fmt]])
```

Fonctions de clustering dans les graphiques

KMeans2D

Le groupe de propriétés **Licence de site** contient des propriétés associées à la licence du système Qlik Sense. Tous les champs sont obligatoires et aucun ne doit être vide.

Propriétés de Licence de site

Nom de propriété	Description
Owner name (Nom du propriétaire)	Nom d'utilisateur du propriétaire du produit Qlik Sense.
Owner organization (Entreprise du propriétaire)	Nom de l'entreprise dont le propriétaire du produit Qlik Sense fait partie.
Serial number (Numéro de série)	Numéro de série du logiciel Qlik Sense.
Control number (Numéro de contrôle)	Numéro de contrôle du logiciel Qlik Sense.
Accès LEF	Le License Enabler File (LEF) du logiciel Qlik Sense.

KMeans2D() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il évalue l'id du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'agorithme sont déterminées par les paramètres coordinate_1, et coordinate_2, respectivement. Ces deux paramètres sont des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters. En option, les données peuvent être normalisées par le paramètre de norme.

```
KMeans2D - fonction de graphique(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

KMeansND() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il évalue l'id du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'algorithme sont déterminées par les paramètres coordinate_1, coordinate_2, etc., jusqu'à n colonnes. Ces paramètres sont tous des agrégations. Le nombre de clusters créés est déterminé par le

paramètre num_clusters.

```
KMeansND - fonction de graphique(num_clusters, num_iter, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il affiche la coordonnée souhaitée du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'agorithme sont déterminées par les paramètres coordinate_1 et coordinate_2, respectivement. Ces deux paramètres sont des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters. En option, les données peuvent être normalisées par le paramètre de norme.

```
KMeansCentroid2D - fonction de graphique(num_clusters, coordinate_no,
coordinate_1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il affiche la coordonnée souhaitée du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'algorithme sont déterminées par les paramètres coordinate_1, coordinate_2, etc., jusqu'à n colonnes. Ces paramètres sont tous des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters.

```
KMeansCentroidND - fonction de graphique(num_clusters, num_iter, coordinate_
no, coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Rank - fonction de graphique

Rank() évalue les lignes du graphique dans l'expression et, pour chaque ligne, affiche la position relative de la valeur de la dimension évaluée dans l'expression. Lors de l'évaluation de l'expression, la fonction compare le résultat à celui des autres lignes contenant le segment de colonne actif et renvoie le classement de la ligne active dans ce segment.

Segments de colonne

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

Dans d'autres graphiques que les tableaux, le segment de colonne actif est défini tel qu'il apparaît dans l'équivalent du tableau simple du graphique.

Syntaxe :

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
mode	Spécifie la représentation numérique du résultat de la fonction.
fmt	Spécifie la représentation textuelle du résultat de la fonction.
TOTAL	Si le graphique est unidimensionnel ou si l'expression est précédée du qualificateur **TOTAL**, l'évaluation de la fonction porte sur la colonne toute entière. Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Le classement est renvoyé sous forme de valeur double, qui est, dans le cas d'un classement unique pour chaque ligne, un entier compris entre 1 et le nombre de lignes dans le segment de colonne actif.

Dans le cas où plusieurs lignes partagent le même classement, il est possible de contrôler la représentation alphanumérique à l'aide des paramètres **mode** et **fmt**.

mode

Le second argument, **mode**, admet les valeurs suivantes :

Exemples de **mode**

Valeur	Description
0 (par défaut)	Si tous les rangs du groupe commun sont inférieurs à la valeur médiane du classement total, toutes les lignes obtiennent le rang le plus bas du groupe. Si tous les rangs du groupe commun sont supérieurs à la valeur médiane du classement total, toutes les lignes obtiennent le rang le plus élevé du groupe. Si les rangs du groupe commun se trouvent de part et d'autre de la valeur médiane, toutes les lignes obtiennent la valeur correspondant à la moyenne du classement supérieur et du classement inférieur du segment de colonne entier.
1	Rang le plus bas sur toutes les lignes.
2	Rang moyen sur toutes les lignes.
3	Rang le plus élevé sur toutes les lignes.
4	Rang le plus bas sur la première ligne, puis incrémenté d'une unité pour chaque ligne.

fmt

Le troisième argument, **fmt**, admet les valeurs suivantes :

Exemples de **fmt**

Valeur	Description
0 (par défaut)	Valeur faible - valeur élevée sur toutes les lignes (par exemple 3 - 4).
1	Valeur faible sur toutes les lignes.
2	Valeur faible sur la première ligne, vide sur les lignes suivantes.

L'ordre des lignes pour le **mode** 4 et le format **fmt** 2 est déterminé par l'ordre de tri des dimensions du graphique.

Exemples et résultats :

Créez deux visualisations à partir des dimensions Product et Sales et une autre à partir de Product et UnitSales. Ajoutez des mesures comme indiqué dans la table suivante.

Exemples de classement

Exemples	Résultats
Exemple 1. Créez une table comportant les dimensions `Customer` et `Sales` et la mesure `Rank(Sales)`.	Le résultat dépend de l'ordre de tri des dimensions. Si la table est triée d'après la dimension Customer, la table répertorie toutes les valeurs de la dimension Sales pour Astrida, puis Betacab, et ainsi de suite. Les résultats de la mesure Rank (Sales) indiquent 10 pour la valeur Sales 12, 9 pour la valeur Sales 13, et ainsi de suite, avec une valeur de rang de 1 renvoyée pour la valeur Sales 78. Le segment de colonne suivant commence par Betacab, pour lequel la première valeur de la dimension Sales dans le segment est égale à 12. La valeur de rang correspondante fournie pour la mesure Rank(Sales) est 11. Si la table est triée sur la base de la dimension Sales, les segments de colonne se composent des valeurs de la dimension Sales et des valeurs Customer correspondantes. Étant donné qu'il y a deux valeurs Sales égales à 12 (pour Astrida et Betacab), la valeur de Rank(Sales) pour ce segment de colonne correspond à 1-2, pour chaque valeur définie sous Customer. Cela s'explique par le fait qu'il y a deux valeurs de dimension Customer pour la valeur de dimension Sales 12. S'il y avait eu 4 valeurs, le résultat serait 1-4, pour toutes les lignes. Cela montre le résultat obtenu avec la valeur par défaut (0) de l'argument fmt.

Exemples	Résultats
Exemple 2. Remplacez la dimension Customer par la dimension Product et ajoutez la mesure `Rank(Sales,1,2)`.	1 est renvoyé sur la première ligne de chaque segment de colonne tandis que toutes les autres lignes restent vides, car les arguments **mode** et **fmt** sont définis sur 1 et 2 respectivement.

Résultats pour l'exemple 1, avec la table triée d'après la dimension Customer :

Table des résultats

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

Résultats pour l'exemple 1, avec la table triée d'après la dimension Sales :

Table des résultats

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

Données utilisées dans les exemples :

```
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 '|');
```

Voir aussi :

p *Sum - fonction de graphique (page 349)*

HRank - fonction de graphique

HRank() évalue l'expression et compare le résultat à celui des autres colonnes contenant le segment de ligne actif d'un tableau croisé dynamique. La fonction renvoie ensuite le classement de la colonne active dans le segment.

Syntaxe :

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Type de données renvoyé : double

Cette fonction ne marche que dans les tableaux croisés dynamiques. Dans tous les autres types de graphique, elle renvoie NULL.

Arguments :

Arguments

Argument	Description
expr	Expression ou champ contenant les données à mesurer.
mode	Spécifie la représentation numérique du résultat de la fonction.
fmt	Spécifie la représentation textuelle du résultat de la fonction.
TOTAL	Si le graphique est unidimensionnel ou si l'expression est précédée du qualificateur **TOTAL**, l'évaluation de la fonction porte sur la colonne toute entière. Si la table ou l'équivalent en tableau comporte plusieurs dimensions verticales, le segment de colonne actif comprend uniquement les lignes contenant les mêmes valeurs que la ligne active dans toutes les colonnes de dimensions, à l'exception de la colonne affichant la dernière dimension dans l'ordre de tri inter-champs.

Si le tableau croisé dynamique est unidimensionnel ou si l'expression est précédée du qualificateur **total**, le segment de ligne actif est toujours égal à la ligne entière. Si le tableau croisé dynamique comporte plusieurs dimensions horizontales, le segment de ligne actif inclura uniquement les colonnes contenant les mêmes valeurs que la colonne active dans toutes les lignes de dimension, à l'exception de la ligne affichant la dernière dimension horizontale dans l'ordre de tri inter-champs.

Le classement est renvoyé sous forme de valeur double, qui sera, dans le cas d'un classement unique pour chaque colonne, un entier compris entre 1 et le nombre de colonnes dans le segment de ligne actif.

Dans le cas où plusieurs lignes partagent le même classement, il est possible de contrôler la représentation alphanumérique à l'aide des arguments **mode** et **format**.

Le deuxième argument **mode** spécifie la représentation numérique du résultat de la fonction :

Exemples de **mode**

Valeur	Description
0 (par défaut)	Si tous les rangs du groupe de partage sont inférieurs à la valeur médiane du classement total, toutes les colonnes obtiennent le rang le plus bas du groupe. Si tous les rangs du groupe de partage sont supérieurs à la valeur médiane du classement total, toutes les colonnes obtiennent le rang le plus élevé du groupe. Si les rangs du groupe commun se trouvent de part et d'autre de la valeur médiane, toutes les lignes obtiennent la valeur correspondant à la moyenne du classement supérieur et du classement inférieur du segment de colonne entier.
1	Rang inférieur de toutes les colonnes du groupe.
2	Rang moyen de toutes les colonnes du groupe.
3	Rang supérieur de toutes les colonnes du groupe.
4	Rang inférieur sur la première colonne, puis incrémenté d'une unité pour chaque colonne du groupe.

Le troisième argument **format** spécifie la représentation textuelle du résultat de la fonction :

Exemples de **format**

Valeur	Description
0 (par défaut)	Valeur inférieure**&' - '&**valeur supérieure sur toutes les colonnes du groupe (par exemple, 3 - 4).
1	Valeur inférieure sur toutes les colonnes du groupe.
2	Valeur inférieure sur la première colonne, vide sur les colonnes suivantes du groupe.

L'ordre des colonnes pour le **mode** 4 et le **format** 2 est déterminé par l'ordre de tri des dimensions du graphique.

Exemples :

```
HRank( sum( Sales ))
HRank( sum( Sales ), 2 )
HRank( sum( Sales ), 0, 1 )
```

Optimisation à l'aide de K-moyennes : Exemple dans le monde réel

L'exemple suivant illustre une utilisation dans le monde réel dans laquelle les fonctions d'algorithme des k-moyennes et de centroïde sont appliquées à un ensemble de données. La fonction K-moyennes sépare les points de données en clusters qui partagent des similarités. Les clusters deviennent plus compacts et

différentiés à mesure que l'algorithme des k-moyennes est appliqué à un nombre configurable d'itérations.

La fonction K-moyennes est utilisée dans de nombreux domaines et dans une grande variété de cas d'utilisation ; parmi les exemples d'utilisation de la mise en cluster figurent la segmentation des clients, la détection des fraudes, la prédiction des attritions de comptes, le ciblage des incitations client, l'identification des cybercrimes et l'optimisation des itinéraires de livraison. L'algorithme des k-moyennes est de plus en plus utilisé dans les entreprises qui cherchent à déduire les tendances et à optimiser les offres de services.

Qlik Sense Fonctions K-moyennes et Centroïde

Qlik Sense fournit deux fonctions K-moyennes qui regroupent les points de données en clusters en fonction de leur similarité. Voir *KMeans2D - fonction de graphique (page 1389)* et *KMeansND - fonction de graphique (page 1402)*. La fonction **KMeans2D** accepte deux dimensions et fonctionne bien pour la visualisation des résultats via un graphique **Nuage de points**. La fonction **KMeansND** accepte plus de deux dimensions. Étant donné qu'il est facile de conceptualiser un résultat en 2D sur des graphiques standard, la démonstration suivante applique la fonction K-moyennes à un graphique **Nuage de points** utilisant deux dimensions. L'algorithme des k-moyennes peut être visualisé via la coloration par expression ; ou par dimension, comme décrit dans cet exemple.

Les fonctions Centroïde Qlik Sense déterminent la position moyenne arithmétique de l'ensemble des points de données du cluster et identifient un point central, ou centroïde, pour ce cluster. Pour chaque ligne (ou chaque enregistrement) du graphique, la fonction de centroïde affiche la coordonnée du cluster auquel ce point de données a été assigné. Voir *KMeansCentroid2D - fonction de graphique (page 1415)* et *KMeansCentroidND - fonction de graphique (page 1417)*.

Vue d'ensemble d'un cas d'utilisation et d'un exemple

L'exemple suivant illustre un scénario simulé dans le monde réel. Une entreprise textile basée dans l'état de New York, aux États-Unis, doit réduire ses dépenses en minimisant les frais de livraison. Une manière d'y parvenir consiste à rapprocher les entrepôts des distributeurs. L'entreprise emploie 118 distributeurs dans l'état de New York. La démonstration suivante simule la manière dont un responsable des opérations peut segmenter les distributeurs en cinq géographies regroupées via la fonction K-moyennes, puis identifier cinq emplacements d'entrepôt optimaux centraux par rapport à ces clusters via la fonction de centroïde. L'objectif est de découvrir les coordonnées de mappage à utiliser pour identifier cinq emplacements d'entrepôt centraux.

L'ensemble de données

L'ensemble de données est basé sur des noms et adresses dans l'état de New York, générés de manière aléatoire avec des coordonnées de latitude et de longitude réelles. L'ensemble de données contient les dix colonnes suivantes : id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. L'ensemble de données est disponible ci-dessous sous forme de fichier que vous pouvez télécharger localement, puis charger dans Qlik Sense ou intégré pour l'éditeur de chargement de données. L'application créée est nommée *Distributeurs - K-moyennes et Centroïde* et la première feuille de l'application est nommée *Analyse des clusters de distribution*.

Sélectionnez le lien suivant pour télécharger le fichier d'échantillon de données : DistributorData.csv

Ensemble de données Distributor : Chargement intégré de l'éditeur de chargement de données dans Qlik Sense (page 1386)

Titre : DistributorData

Nombre total d'enregistrements : 118

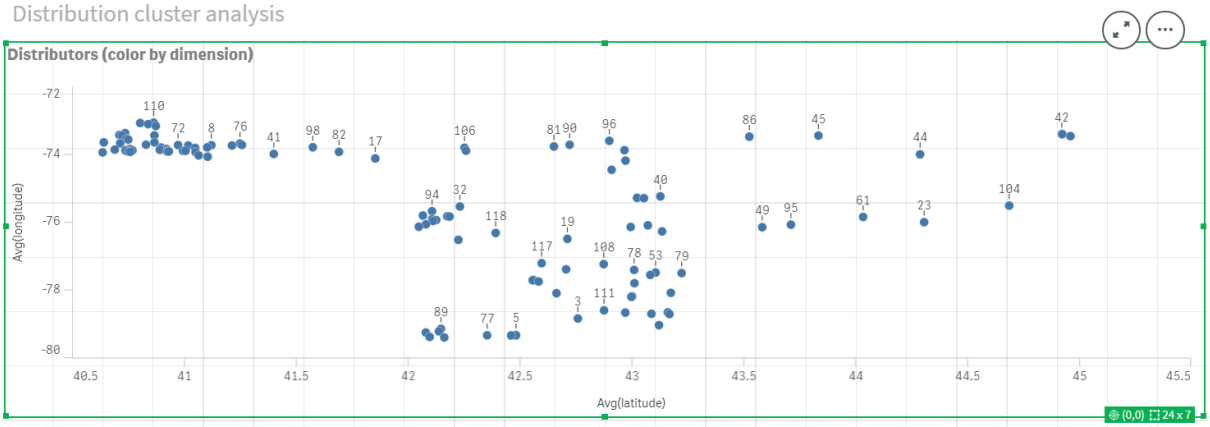
Application de la fonction KMeans2D

Dans cet exemple, la configuration d'un graphique **Nuage de points** est démontrée via l'ensemble de données *DistributorData*, la fonction **KMeans2D** est appliquée et le graphique est coloré par dimension.

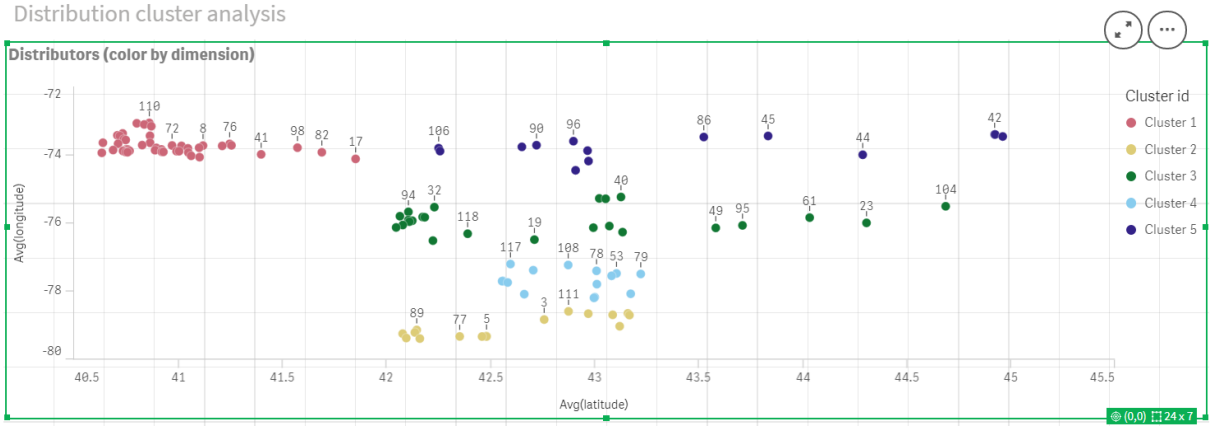
Notez que les fonctions K-moyennes Qlik Sense prennent en charge le clustering automatique via une méthode dite Différence de profondeur (DeD - Depth Difference). Quand un utilisateur définit 0 comme nombre de clusters, le nombre optimal de clusters est déterminé pour cet ensemble de données. Pour cet exemple, cependant, une variable est créée pour l'argument **num_clusters** (voir *KMeans2D - fonction de graphique (page 1389)* pour la syntaxe). Par conséquent, le nombre de clusters souhaité (k=5) est spécifié par une variable.

1. Un graphique **Nuage de points** est glissé sur la feuille et nommé *Distributeurs (par dimension)*.
2. Une **variable** est créée pour spécifier le nombre de clusters. La **variable** est nommée *vDistClusters*. Pour la variable **Définition**, saisissez *5*.
3. Configuration des **Données** pour le graphique :
    a. sous **Dimensions**, le champ *id* est sélectionné pour **Bulle**. *Id de cluster* est la valeur saisie pour **Étiquette**.
    b. Sous **Mesures**, *Avg([latitude])* est l'expression pour **Axe des abscisses**.
    c. Sous **Mesures**, *Avg([longitude])* est l'expression pour **Axe des ordonnées**.
4. Configuration de l'**Apparence** :
    a. Sous **Couleurs et légende**, **Personnalisées** est sélectionné pour **Couleurs**.
    b. **Par dimension** est sélectionné pour la coloration du graphique.
    c. L'expression suivante est saisie : *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*.
    d. La case **Couleurs persistantes** est cochée.

Nuage de points avant K-moyennes, coloration par dimension appliquée



Nuage de points après coloration K-moyennes par dimension est appliqué



.

Ajout d'une **table** : *Distributeurs*

Il peut s'avérer utile d'avoir une table prête pour accéder rapidement aux données pertinentes. Le graphique **Nuage de points** affiche les *ID* dans une table avec le nom du distributeur correspondant ajouté pour référence.

1. Une **table** nommée *Distributeurs* est glissée sur la feuille avec les **Colonnes** (Dimensions) suivantes : *id*, *first_name* et *last_name*.

Table : Nom des distributeurs

Distributors

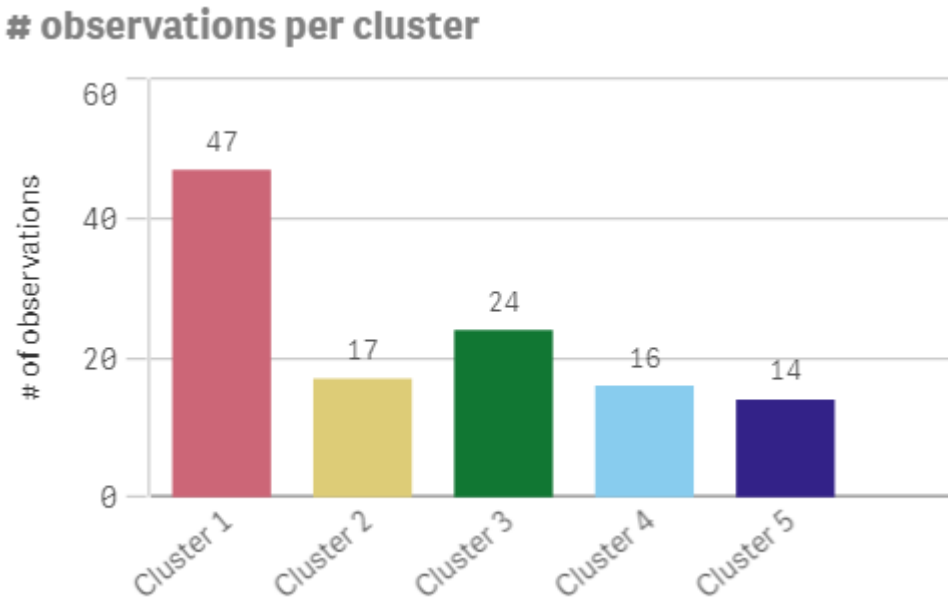
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Ajout d'un **graphique en barres** : *Nbre d'observations par cluster*

Pour le scénario de distribution des entrepôts, il est utile de connaître le nombre de distributeurs qui seront servis par chaque entrepôt. Par conséquent, on crée un **graphique en barres** qui mesure le nombre de distributeurs assignés à chaque cluster.

1. Un **graphique en barres** est glissé sur la feuille. Le graphique est nommé : *Nbre d'observations par cluster*.
2. Configuration des **données** pour le **graphique en barres** :
    a. Une **Dimension** étiquetée *Clusters* est ajoutée (il est possible d'ajouter l'étiquette après l'application de l'expression). L'expression suivante est saisie : *=pick(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*.
    b. Une **Mesure** étiquetée *Nbre d'observations* est ajoutée. L'expression suivante est saisie : *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*.
3. Configuration de l'**Apparence** :
    a. Sous **Couleurs et légende**, **Personnalisées** est sélectionné pour **Couleurs**.
    b. **Par dimension** est sélectionné pour la coloration du graphique.
    c. L'expression suivante est saisie : *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*.
    d. La case **Couleurs persistantes** est cochée.
    e. **Afficher la légende** est désactivé.
    f. Sous **Présentation**, **Étiquettes de valeur** est basculé sur **Auto**.
    g. Sous **Axe des abscisses** : **Clusters**, **Étiquettes uniquement** est sélectionné.

Graphique en barres : Nbre d'observations par cluster



Application de la fonction **Centroid2D**

Une deuxième table est ajoutée pour la fonction **Centroid2D** qui identifiera les coordonnées des emplacements d'entrepôts potentiels. Cette table affiche l'emplacement central (les valeurs de centroïde) des cinq groupes de distributeurs identifiés.

1. Une **Table** est glissée sur la feuille et nommée *Centroïdes de clusters* avec les colonnes suivantes :
    a. Une **Dimension** étiquetée *Clusters* est ajoutée. L'expression suivante est saisie :*=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*).
    b. Une **Mesure** étiquetée *latitude (D1)* est ajoutée. L'expression suivante est saisie :*=only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*.
    Notez que le paramètre **coordinate_no** correspond à la première dimension(0). Dans ce cas, la dimension *latitude* est tracée par rapport à l'axe des abscisses. Si nous travaillions avec la fonction **CentroidND** et s'il existait jusqu'à six dimensions, ces entrées de paramètres pourraient prendre n'importe laquelle des six valeurs : 0, 1, 2, 3, 4 ou 5.
    c. Une **Mesure** étiquetée *longitude (D2)* est ajoutée. L'expression suivante est saisie :*=only (aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*.
    Le paramètre **coordinate_no** de cette expression correspond à la deuxième dimension(1). La dimension *longitude* est tracée par rapport à l'axe des ordonnées.

Table : Calcul des centroïdes des clusters

Cluster centroids

Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Mappage des centroïdes

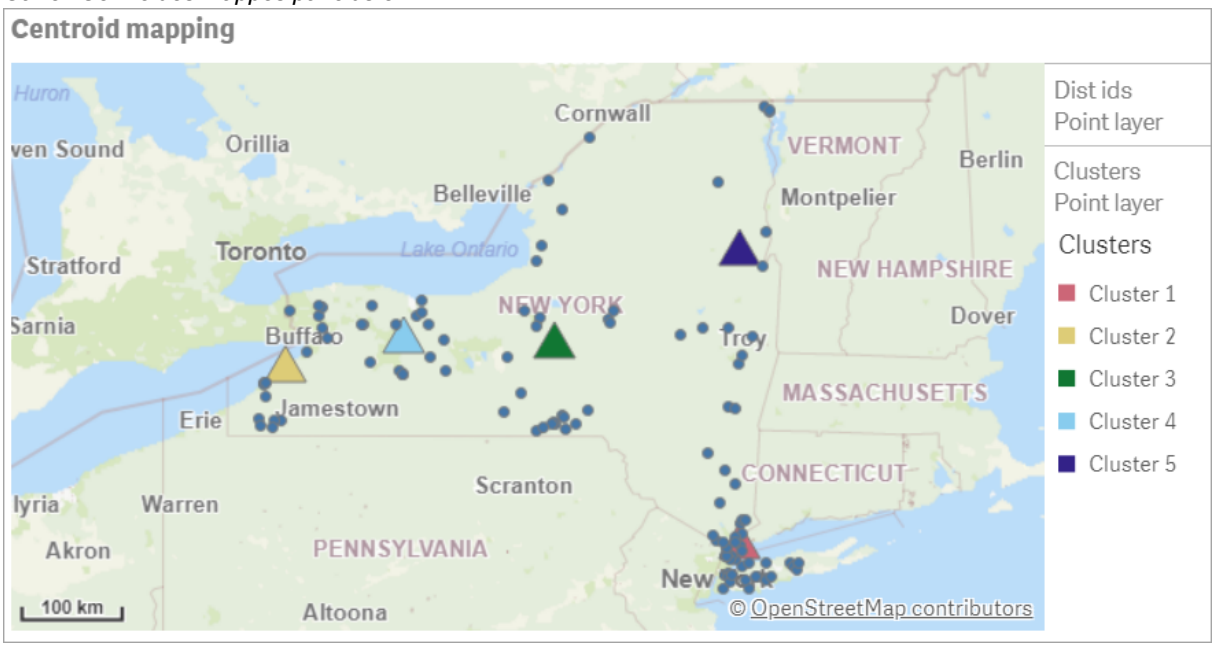
L'étape suivante consiste à mapper les centroïdes. C'est au développeur d'applications de décider s'il préfère placer la visualisation sur des feuilles distinctes.

1. Une **carte** nommée *Mappage des centroïdes* est glissée sur la feuille.
2. Dans la section **Couches**. **Ajouter une couche** est sélectionné, suivi de **Couche de points**.
    a. Le **champ** *id* est sélectionné et l'**étiquette** *Dist ids* est ajoutée.
    b. Dans la section **Emplacement**, la case **Champs de latitude et de longitude** est cochée.
    c. Pour **Latitude**, le champ *latitude* est sélectionné.
    d. Pour **Longitude**, le champ *longitude* est sélectionné.
    e. Dans la section **Taille et forme**, **Bulle** est sélectionné pour **Forme** et la **Taille** est réduite en fonction des préférences à l'aide du curseur.

f. Dans la section **Couleurs**, **Couleur unique** est sélectionné et la couleur bleue est sélectionnée pour **Couleur** et la couleur grise pour **Couleur du contour** (ces choix dépendent eux aussi des préférences).

3. Dans la section **Couches**, une deuxième **Couche de points** est ajoutée en sélectionnant **Ajouter une couche**, puis **Couche de points**.
   a. L'expression suivante est saisie : *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*.
   b. L'**Étiquette** *Clusters* est ajoutée.
   c. Dans la section **Emplacement**, la case **Champs de latitude et de longitude** est cochée.
   d. Pour **Latitude**, qui, dans ce cas, est tracé le long de l'axe des abscisses, l'expression suivante est ajoutée : *=aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*.
   e. Pour **Longitude**, qui, dans ce cas, est tracé le long de l'axe des ordonnées, l'expression suivante est ajoutée : *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*.
   f. Dans la section **Taille et forme**, **Triangle** est sélectionné pour **Forme** et la **Taille** est réduite selon les préférences à l'aide du curseur.
   g. Sous **Couleurs et légende**, **Personnalisées** est sélectionné pour **Couleurs**.
   h. **Par dimension** est sélectionné pour la coloration du graphique. L'expression suivante est saisie : *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   i. La dimension est étiquetée *Clusters*.
4. Dans **Paramètres de la carte**, **Adaptative** est sélectionné pour **Projection**. **Métriques** est sélectionné pour **Unités de mesurement**.

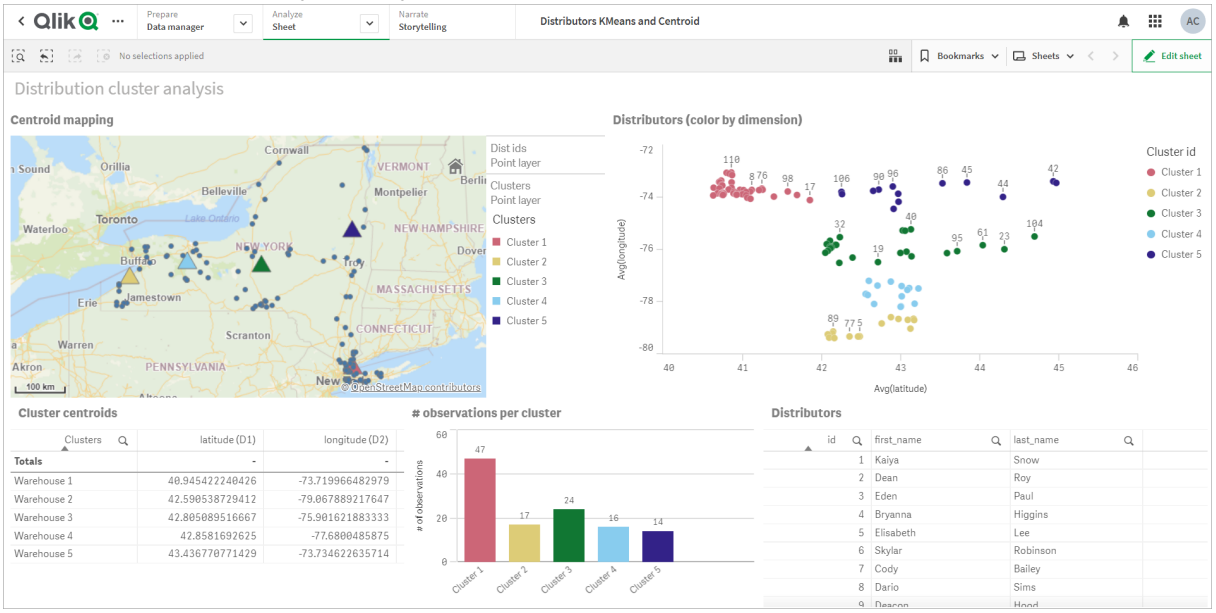
Carte : Centroïdes mappés par cluster



Conclusion

Grâce à la fonction K-moyennes utilisé dans ce scénario dans le monde réel, les distributeurs ont été segmentés en groupes ou clusters similaires en fonction de leur similarité ; dans ce cas, la proximité des uns par rapport aux autres. La fonction Centroïde a été appliquée à ces clusters pour identifier cinq coordonnées de mappage. Ces coordonnées fournissent un emplacement central initial auquel construire ou placer les entrepôts. La fonction Centroïde est appliquée au graphique **Carte** pour que les utilisateurs de l'application puissent visualiser l'emplacement des centroïdes par rapport aux points de données de clusters environnants. Les coordonnées obtenues représentent les emplacements d'entrepôts potentiels susceptibles de minimiser les frais de livraison aux distributeurs dans l'état de New York.

Application : Exemple d'analyse K-moyennes et Centroïde



Ensemble de données Distributor : Chargement intégré de l'éditeur de chargement de données dans Qlik Sense

```
DistributorData: Load * Inline [ id,first_name,last_
name,telephone,address,city,state,zip,latitude,longitude 1,Kaiya,Snow,(716) 201-1212,6231
Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-78.69313 2,Dean,Roy,(716) 201-
1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036 3,Eden,Paul,(716) 202-4596,4647
Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194 4,Bryanna,Higgins,(716) 203-
7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088 5,Elisabeth,Lee,(716) 203-7043,36 E
Courtney St,Dunkirk,NY,14048,42.48299,-79.31928 6,Skylar,Robinson,(716) 203-7166,26 Greco
Ln,Dunkirk,NY,14048,42.4612095,-79.3317925 7,Cody,Bailey,(716) 203-7201,114 Lincoln
Ave,Dunkirk,NY,14048,42.4801269,-79.322232 8,Dario,Sims,(408) 927-1606,N Castle
Dr,Armonk,NY,10504,41.11979,-73.714864 9,Deacon,Hood,(410) 244-6221,4856 44th
St,Woodside,NY,11377,40.748372,-73.905445 10,Zackery,Levy,(410) 363-8874,61 Executive
Blvd,Farmingdale,NY,11735,40.7197457,-73.430239 11,Rey,Hawkins,(412) 344-8687,4585 Shimerville
Rd,Clarence,NY,14031,42.972075,-78.6592452 12,Phillip,Howard,(413) 269-4049,464 Main St
#101,Port Washington,NY,11050,40.8273756,-73.7009971 13,Shirley,Tyler,(434) 985-8943,114 Glann
Rd,Apalachin,NY,13732,42.0482515,-76.1229725 14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown
```

Rd,Pearl River,NY,10965,41.0629,-74.0159 15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West Harrison,NY,10604,41.0162722,-73.7234926 16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New Hartford,NY,13413,43.0555739,-75.2793197 17,Harper,Gibbs,(239) 466-0238,Po Box 33,Cottekill,NY,12419,41.853392,-74.106082 18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill Rd,East Syracuse,NY,13057,43.073215,-76.081448 19,Roberto,Wade,(270) 469-1211,3936 Holley Rd,Moravia,NY,13118,42.713044,-76.481227 20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte #3,Naples,NY,14512,42.707366,-77.380489 21,Dale,Andersen,(281) 480-5690,205 W Service Rd,Champlain,NY,12919,44.9645392,-73.4470831 22,Lorelai,Burch,(302) 644-2133,1 Brewster St,Glen Cove,NY,11542,40.865177,-73.633019 23,Amiyah,Flowers,(303) 223-0055,46600 Us Interstate 81 Rte,Alexandria Bay,NY,13607,44.309626,-75.988365 24,Mckinley,Clements,(303) 918-3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298 25,Marc,Gibson,(607) 203-1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614 26,Kali,Norman,(607) 203-1400,1 Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026 27,Laci,Cain,(607) 203-1437,16 Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627 28,Mohammad,Perez,(607) 203-1652,71 Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-75.952187 29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074 30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917 31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124 32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674 33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176 34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602 35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561 36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506 37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651 38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185 39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591 40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726 41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003 42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373 43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884 44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474 45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155 46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079 47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848 48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957 49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317 50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487 51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285 52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452 53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552 54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088 55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983 56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-77.741648 57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661 58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465 59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858 60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997 61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437 62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331 63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029 64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715 65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839 66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555 67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957 68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886 69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-73.879748 70,Blaze,Medina,(914) 207-0015,60 Elliott

```
Ave,Yonkers,NY,10705,40.921498,-73.896682 71,Finnegan,Tucker,(914) 207-0015,90 Hillside
Drive,Yonkers,NY,10705,40.922514,-73.892911 72,Pranav,Palmer,(914) 214-8376,5 Bruce
Ave,Harrison,NY,10528,40.970916,-73.711493 73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount
Kisco,NY,10549,41.211993,-73.723202 74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs
Ferry,NY,10522,41.004629,-73.879825 75,Lamar,Pierce,(914) 232-0380,68 Ridge
Rd,Katonah,NY,10536,41.256662,-73.707964 76,Carla,Coffey,(914) 232-0469,197 Beaver Dam
Rd,Katonah,NY,10536,41.247934,-73.664363 77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow
Rd,Cassadega,NY,14718,42.353861,-79.329558 78,Raquel,Hodges,(585) 398-8125,809 County Road
,Victor,NY,14564,43.011745,-77.398806 79,Jerimiah,Gardner,(585) 787-9127,809 Houston
Rd,Webster,NY,14580,43.224204,-77.491353 80,Clarence,Hammond,(720) 746-1619,809 Pierpont
Ave,Piermont,NY,10968,41.0491181,-73.918622 81,Rhys,Gill,(518) 427-7887,81 Columbia
St,Albany,NY,12210,42.652824,-73.752096 82,Edith,Parrish,(845) 452-7621,81 Glenwood
Ave,Poughkeepsie,NY,12603,41.691058,-73.910829 83,Kobe,Mcintosh,(845) 371-1101,81 Heitman
Dr,Spring Valley,NY,10977,41.103227,-74.054396 84,Ayden,Waters,(516) 796-2722,81 Kingfisher
Rd,Levittown,NY,11756,40.738939,-73.52826 85,Francis,Rogers,(631) 427-7728,81 Knollwood
Ave,Huntington,NY,11743,40.864905,-73.426107 86,Jaden,Landry,(716) 496-4038,12839 39th
Rte,Chaffee,NY,14030,43.527396,-73.462786 87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga
Rd,Ballston Spa,NY,12020,42.968594,-73.862847 88,Eduardo,Contreras,(716) 285-8987,1285
Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-79.029274 89,Gabriela,Davidson,(716) 267-
3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976 90,Evangeline,Case,(518) 272-9435,1287
2nd Ave,Watervliet,NY,12189,42.723132,-73.703818 91,Tyrone,Ellison,(518) 843-4691,1287 Midline
Rd,Amsterdam,NY,12010,42.9730876,-74.1700608 92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens
Rd,Athens,NY,12015,42.259381,-73.876897 93,Londyn,Butler,(518) 922-7095,129 Argersinger
Rd,Fultonville,NY,12072,42.910969,-74.441917 94,Graham,Becker,(607) 655-1318,129 Baker
Rd,Windsor,NY,13865,42.107271,-75.66408 95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90
Rte,Mannsville,NY,13661,43.713443,-76.06232 96,Grant,Hoover,(518) 692-8363,1718 County 113
Rte,Schaghticote,NY,12154,42.900648,-73.585036 97,Mark,Goodwin,(631) 584-6761,172 Cambon
Ave,Saint James,NY,11780,40.871152,-73.146032 98,Deacon,Cantu,(845) 221-7940,172 Carpenter
Rd,Hopewell Junction,NY,12533,41.57388,-73.77609 99,Tristian,Walsh,(516) 997-4750,172 E Cabot
Ln,Westbury,NY,11590,40.7480397,-73.54819 100,Abram,Alexander,(631) 588-3817,172 Lorenzo
Cir,Ronkonkoma,NY,11779,40.837123,-73.09367 101,Lesly,Bush,(516) 489-3791,172 Nassau
Blvd,Garden City,NY,11530,40.71147,-73.660753 102,Pamela,Espinoza,(716) 201-1520,172 Niagara
St ,Lockport,NY,14094,43.169871,-78.70093 103,Bryanna,Newton,(914) 328-4332,172 Warren
Ave,White Plains,NY,10603,41.047207,-73.79572 104,Marcelo,Schmitt,(315) 393-4432,319 Mansion
Ave,Ogdensburg,NY,13669,44.690246,-75.49992 105,Layton,Valenzuela,(631) 676-2113,319
Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993 106,Roderick,Rocha,(518) 671-6037,319
Warren St,Hudson,NY,12534,42.252527,-73.790629 107,Camryn,Terrell,(315) 635-1680,3192 Olive
Dr,Baldinsville,NY,13027,43.136843,-76.260303 108,Summer,Callahan,(585) 394-4195,3192 Smith
Road,Canandaigua,NY,14424,42.875457,-77.228039 109,Pierre,Novak,(716) 665-2524,3194 Falconer
Kimball Stand Rd,Falconer,NY,14733,42.138439,-79.211091 110,Kennedi,Fry,(315) 543-2301,32
College Rd,Selden,NY,11784,40.861624,-73.04757 111,Wyatt,Pruitt,(716) 681-4042,277 Ransom
Rd,Lancaster ,NY,14086,42.87702,-78.591302 112,Lilly,Jensen,(631) 841-0859,2772 Schliegel
Blvd,Amityville,NY,11701,40.708021,-73.413015 113,Tristin,Hardin,(631) 920-0927,278 Fulton
Street,West Babylon,NY,11704,40.733578,-73.357321 114,Tanya,Stafford,(716) 484-0771,278
Sampson St,Jamestown,NY,14701,42.0797,-79.247805 115,Paris,Cordova,(607) 589-4857,278 Washburn
Rd,Spencer,NY,14883,42.225046,-76.510257 116,Alfonso,Morse,(718) 359-5582,200 Colden
St,Flushing,NY,11355,40.750403,-73.822752 117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill
Rd,Branchport,NY,14418,42.597957,-77.199267 118,Iris,Wolf,(607) 539-7288,444 Harford
Rd,Brooktondale,NY,14817,42.392164,-76.30756 ];
```

KMeans2D - fonction de graphique

KMeans2D() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il évalue l'id du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'agorithme sont déterminées par les paramètres coordinate_1, et coordinate_2, respectivement. Ces deux paramètres sont des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters. En option, les données peuvent être normalisées par le paramètre de norme.

KMeans2D renvoie une valeur par point de données. La valeur renvoyée est une valeur double et est la valeur d'entier correspondant au cluster auquel chaque point de données a été affecté.

Syntaxe :

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
num_clusters	Entier qui spécifie le nombre de clusters.
coordinate_1	L'agrégation calcule la première coordonnée, généralement l'axe x du nuage de points qui peut être obtenu à partir du graphique. Le paramètre supplémentaire, coordinate_2, calcule la deuxième coordonnée.
norm	La méthode de normalisation optionnelle est appliquée aux ensembles de données avant le clustering KMeans. Valeurs possibles : 0 ou ‘none’ pour aucune normalisation 1 ou ‘zscore’ pour la normalisation z-score 2 ou ‘minmax’ pour la normalisation min-max Si aucun paramètre n'est fourni ou si le paramètre fourni est incorrect, aucune normalisation n'est appliquée. Z-score normalise les données en fonction d'une moyenne des fonctions et d'un écart-type standard. Z-score ne garantit pas que chaque fonction a la même échelle, mais il s'agit d'une meilleure approche que min-max pour traiter les valeurs hors norme. La normalisation min-max garantit que les fonctions ont la même échelle en prenant les valeurs minimale et maximale et chacune et en recalculant chaque point de données.

Exemple : Expression de graphique
Dans cet exemple, nous créons un graphique Nuage de points à l'aide de l'ensemble de données *Iris*, puis nous utilisons KMeans pour colorer les données par expression.

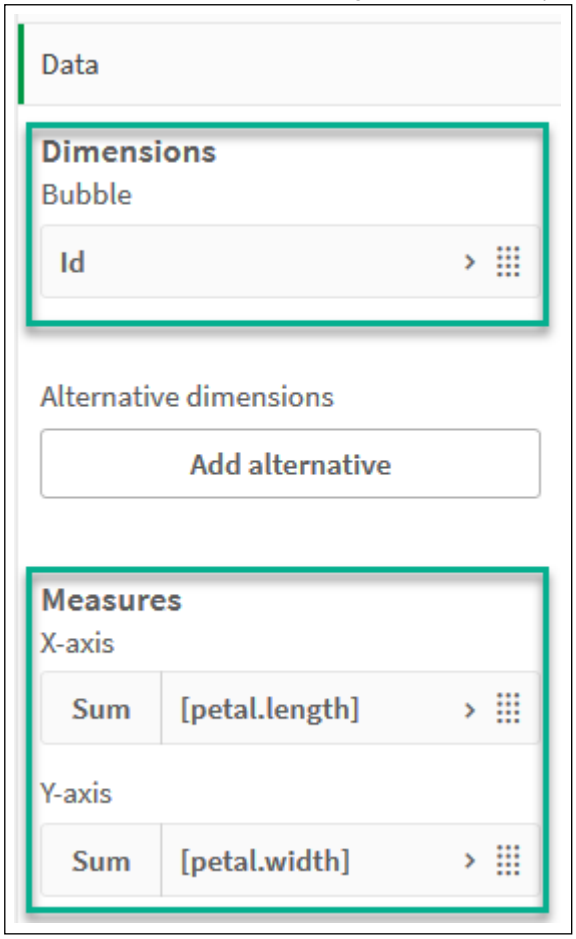
Nous créons également une variable pour l'argument *num_clusters*, puis nous utilisons une zone d'entrée de variable pour modifier le nombre de clusters.

L'ensemble de données *Iris* est publiquement disponible dans une variété de formats. Nous avons fourni les données sous forme de tableau intégré à charger via l'éditeur de chargement de données dans Qlik Sense. Notez que nous avons ajouté une colonne *Id* à la table de données pour cet exemple.

Après avoir chargé les données dans Qlik Sense, procédez comme suit :

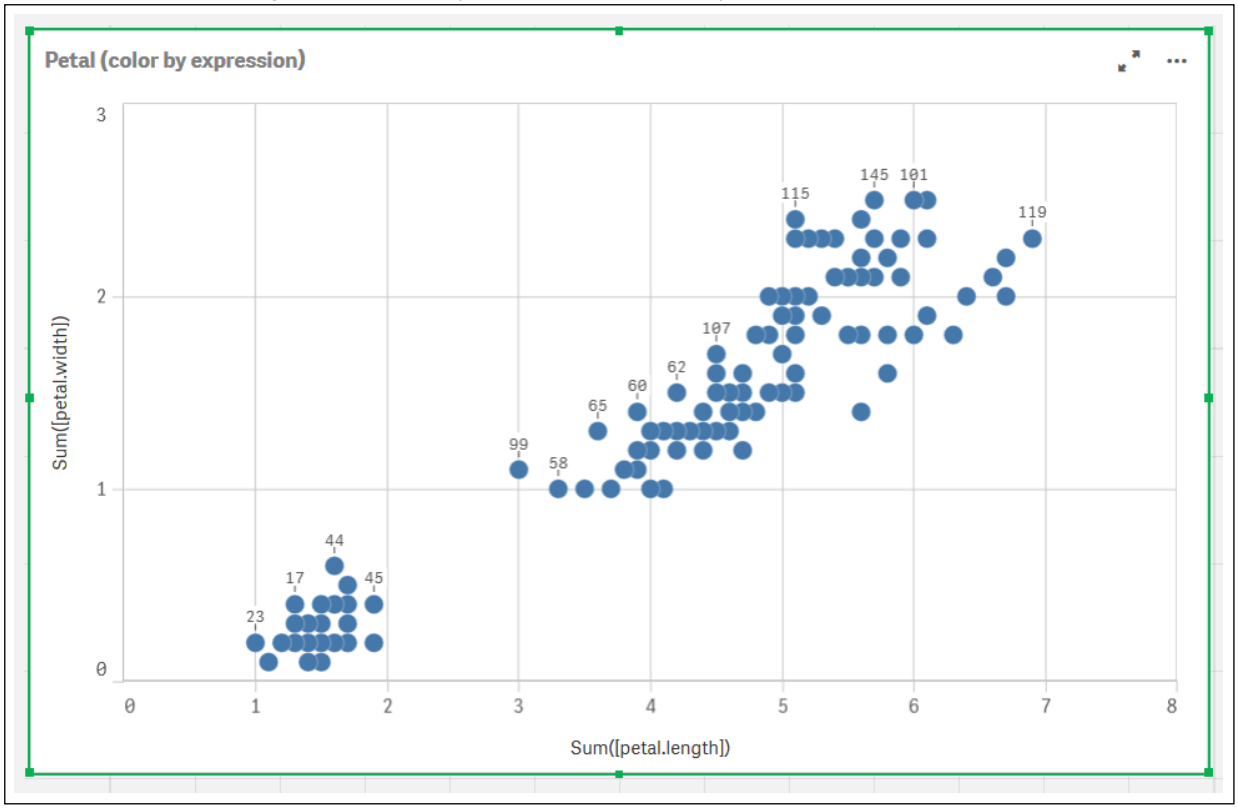
1. Glissez un graphique **Nuage de points** sur une nouvelle feuille. Nommez le graphique *Pétale (expression de la couleur)*.
2. Créez une variable pour spécifier le nombre de clusters. Pour la variable **Nom**, saisissez *KmeansPetalClusters*. Pour la variable **Définition**, saisissez *=2*.
3. Configurez **Données** pour le graphique :
    i. sous **Dimensions**, sélectionnez *id* pour le champ **Bulle**. Saisissez l'Id de cluster de l'Étiquette.
    ii. Sous **Mesures**, sélectionnez *Sum([petal.length])* pour l'expression **Axe X**.
    iii. Sous **Mesures**, sélectionnez *Sum([petal.width])* pour l'expression **Axe Y**.

Paramètres des données pour le graphique Pétale (expression de la couleur)



Les points de données sont tracés sur le graphique.

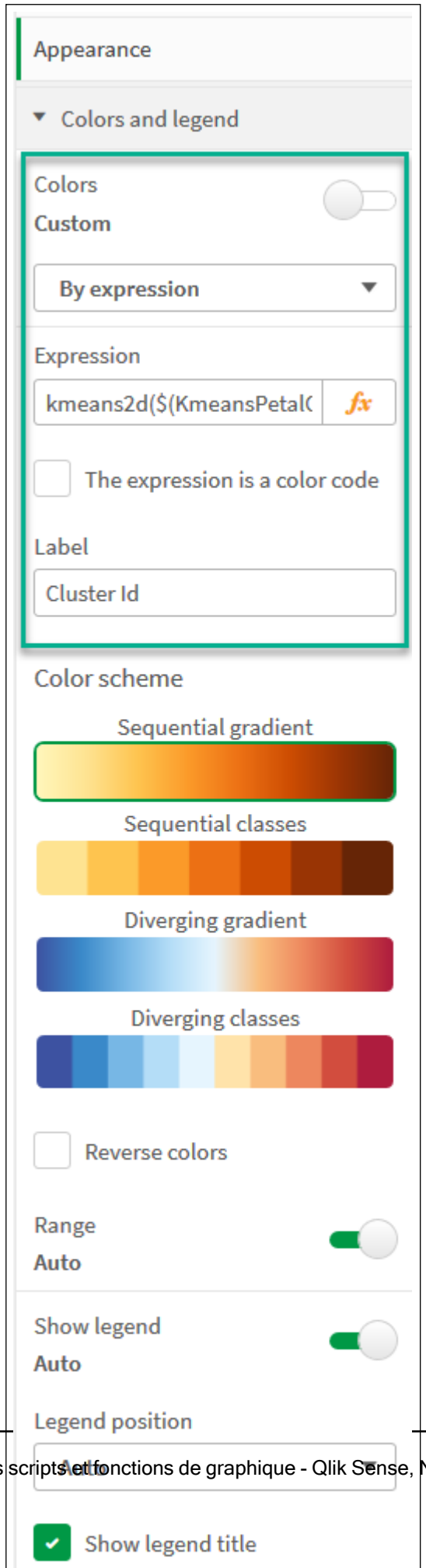
Points de données sur le graphique Pétale (expression de la couleur)



4. Configurez **Aspect** pour le graphique :
   i. Sous **Couleurs et légende**, sélectionnez **Personnaliser** pour **Couleurs**.
   ii. Choisissez de colorer le graphique **Par expression**.
   iii. Saisissez la valeur suivante pour **Expression** : *kmeans2d($(KmeansPetalClusters), Sum ([petal.length]), Sum([petal.width]))*
   Notez que *KmeansPetalClusters* est la variable définie sur *2*.
   Sinon, saisissez la valeur suivante : *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
   iv. Décochez la case **Expression sous forme de code couleur**.

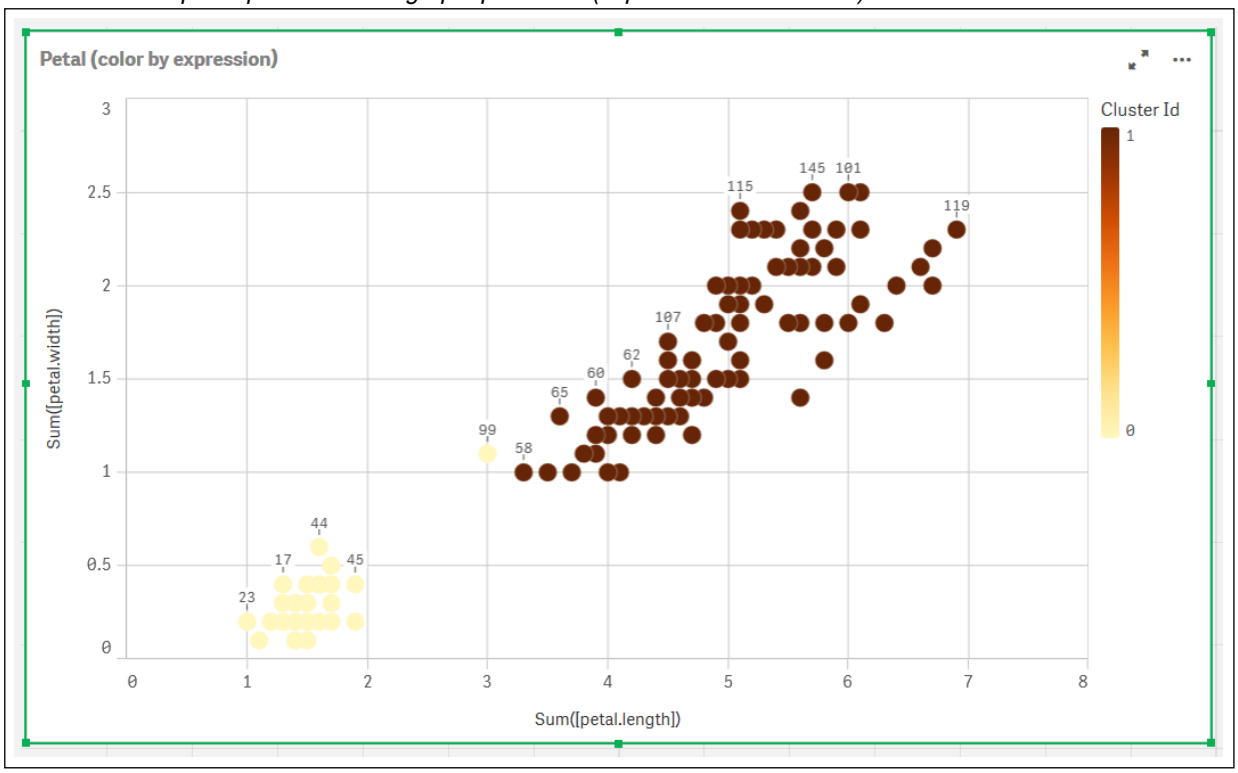
v. Saisissez la valeur suivante pour **Étiquette** : *Id de cluster*

Paramètres d'aspect pour le graphique Pétale (expression de la couleur)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Auto
Show legend title

Les deux clusters du graphique sont colorés par l'expression KMeans.

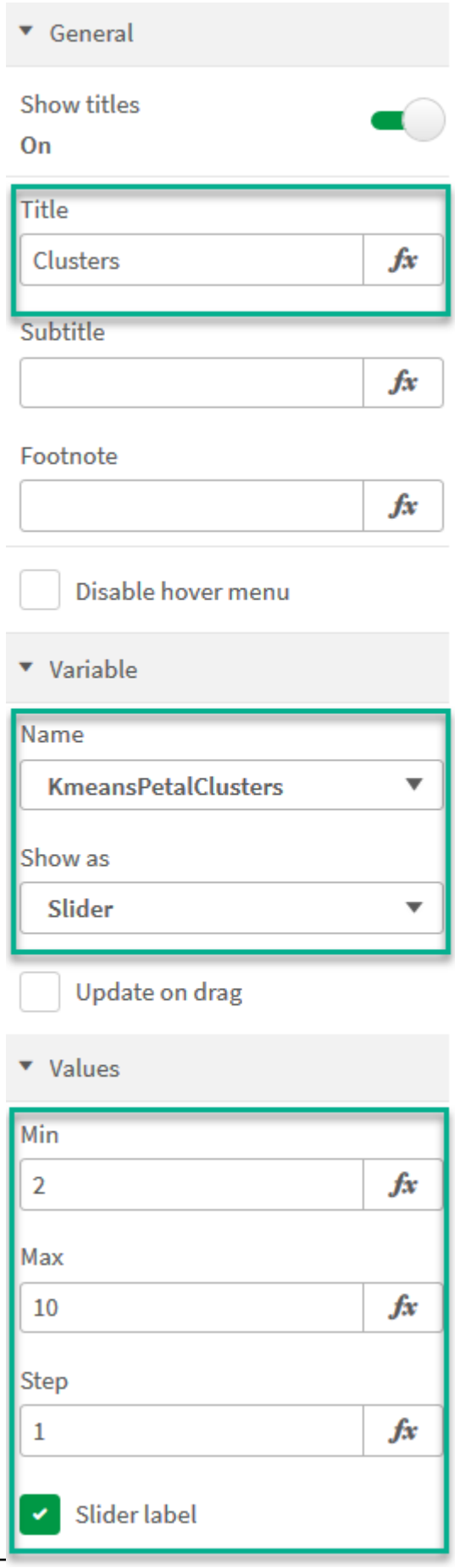
Clusters colorés par expression sur le graphique Pétale (expression de la couleur)



5. Ajoutez une zone **Entrée de variable** pour le nombre de clusters.
    i. Sous **Objets personnalisés** dans le panneau des **Ressources**, sélectionnez **Qlik Dashboard bundle**. Si nous n'avions pas accès au Dashboard bundle, nous pourrions tout de même modifier le nombre de clusters à l'aide de la variable créée, ou directement sous forme d'entier dans l'expression.
    ii. Glissez une zone **Entrée de variable** sur la feuille.
    iii. Sous **Aspect**, cliquez sur **Général**.
    iv. Saisissez la valeur suivante pour **Titre** : *Clusters*
    v. Cliquez sur **Variable**.
    vi. Sélectionnez la variable suivante pour **Nom** : *KmeansPetalClusters*.
    vii. Sélectionnez **Curseur** pour **Afficher comme**.

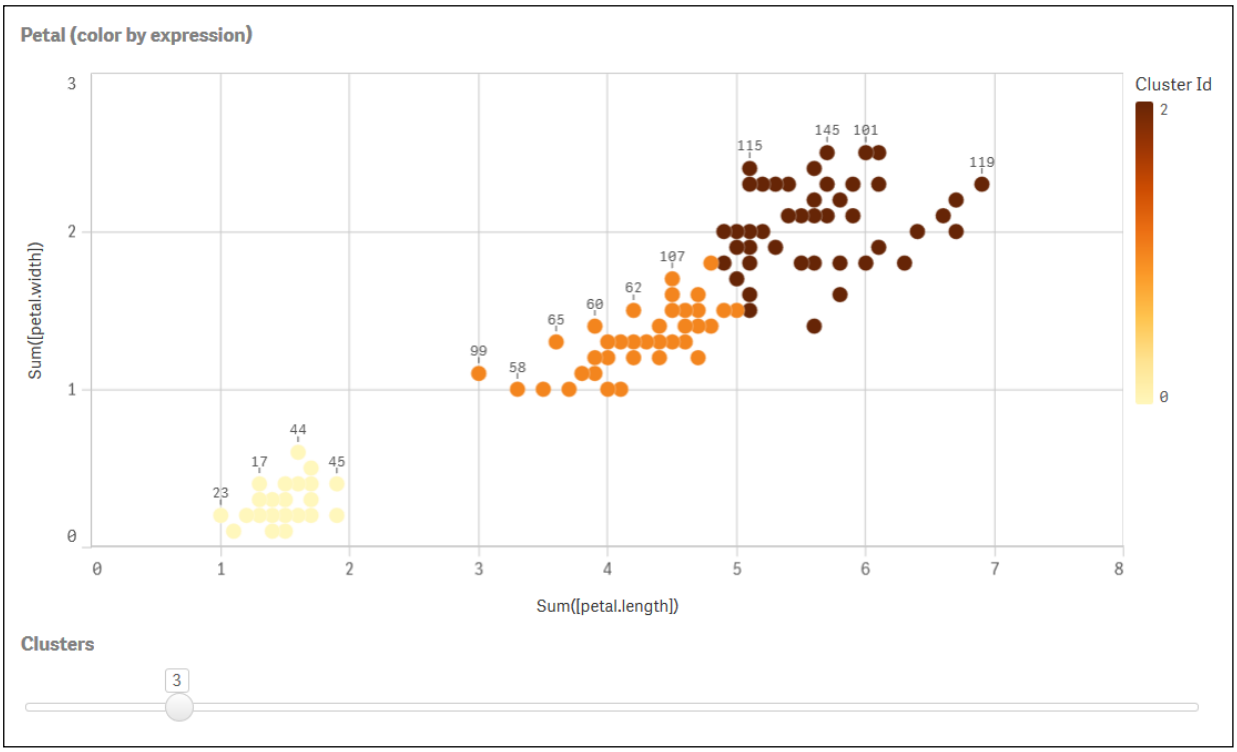
viii. Sélectionnez **Valeurs** et configurez les paramètres selon les besoins.

Aspect pour la zone d'entrée de variable Clusters


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

Une fois l'édition terminée, nous pouvons modifier le nombre de clusters à l'aide du curseur de la zone d'entrée de la variable *Clusters*.

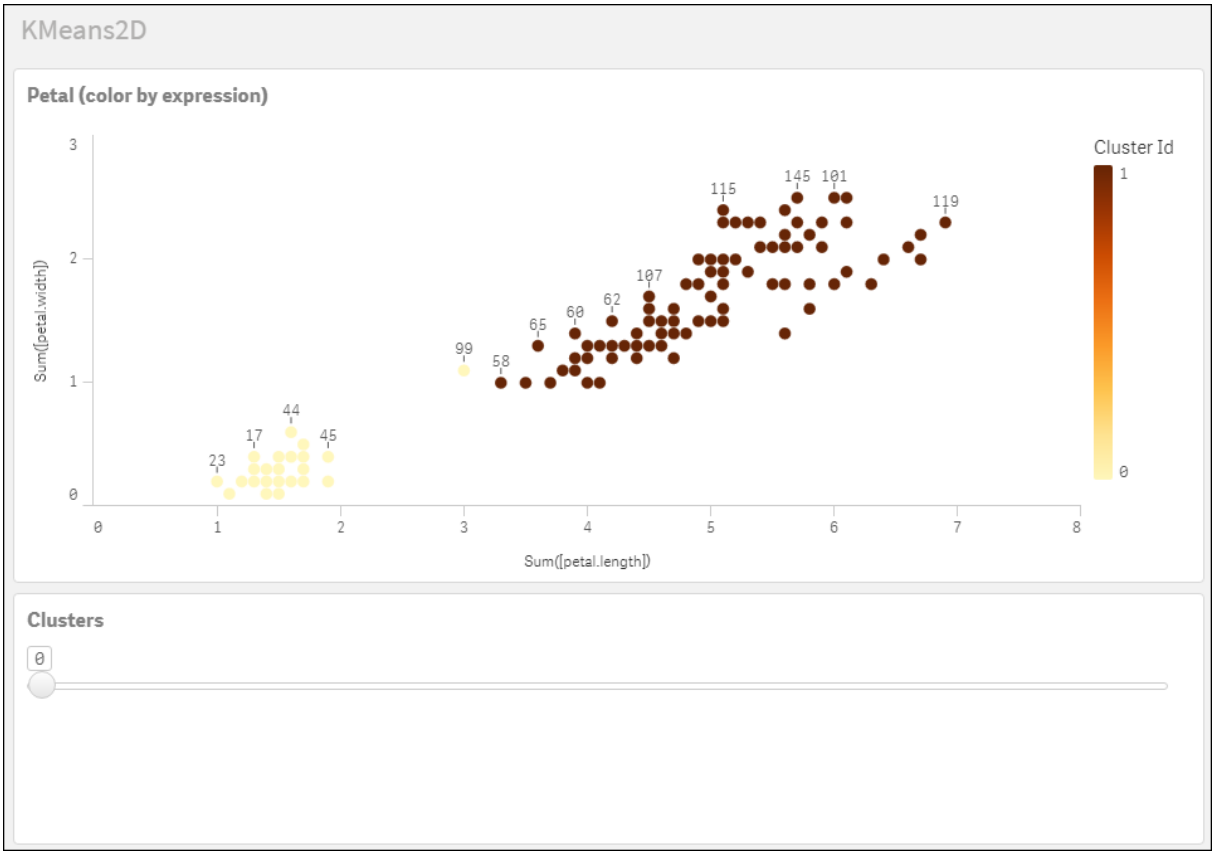
Clusters colorés par expression sur le graphique Pétale (expression de la couleur)



Clustering automatique

Les fonctions **K-moyennes** prennent en charge le clustering automatique via une méthode dite Différence de profondeur (DeD - Depth Difference). Quand un utilisateur définit 0 comme nombre de clusters, un nombre optimal de clusters est déterminé pour cet ensemble de données. Notez que même si entier n'est pas explicitement renvoyé pour le nombre de clusters (*k*), il est calculé dans l'algorithme K-moyennes. Par exemple, si 0 est spécifié dans la fonction pour la valeur de *KmeansPetalClusters* ou défini via une zone d'entrée de variable, les affectations de clusters sont automatiquement calculées pour l'ensemble de données en fonction d'un nombre optimal de clusters.

La méthode Différence de profondeur de K-moyennes détermine le nombre optimal de clusters quand (k) est défini sur 0.



Ensemble de données Iris : Chargement intégré de l'éditeur de chargement de données dans Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
```

```
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansND - fonction de graphique

KMeansND() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il évalue l'id du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'algorithme sont déterminées par les paramètres coordinate_1, coordinate_2, etc., jusqu'à n colonnes. Ces paramètres sont tous des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters.

KMeansND renvoie une valeur par point de données. La valeur renvoyée est une valeur double et est la valeur d'entier correspondant au cluster auquel chaque point de données a été affecté.

Syntaxe :

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
num_clusters	Entier qui spécifie le nombre de clusters.
num_iter	Le nombre d'itérations de clustering avec des centres de cluster réinitialisés.
coordinate_1	L'agrégation calcule la première coordonnée, généralement l'axe x (d'un nuage de points qui peut être obtenu à partir du graphique). Les paramètres supplémentaires calculent les deuxième, troisième et quatrième coordonnées, etc.

Exemple : Expression de graphique
Dans cet exemple, nous créons un graphique Nuage de points à l'aide de l'ensemble de données *Iris*, puis nous utilisons KMeans pour colorer les données par expression.

Nous créons également une variable pour l'argument *num_clusters*, puis nous utilisons une zone d'entrée de variable pour modifier le nombre de clusters.

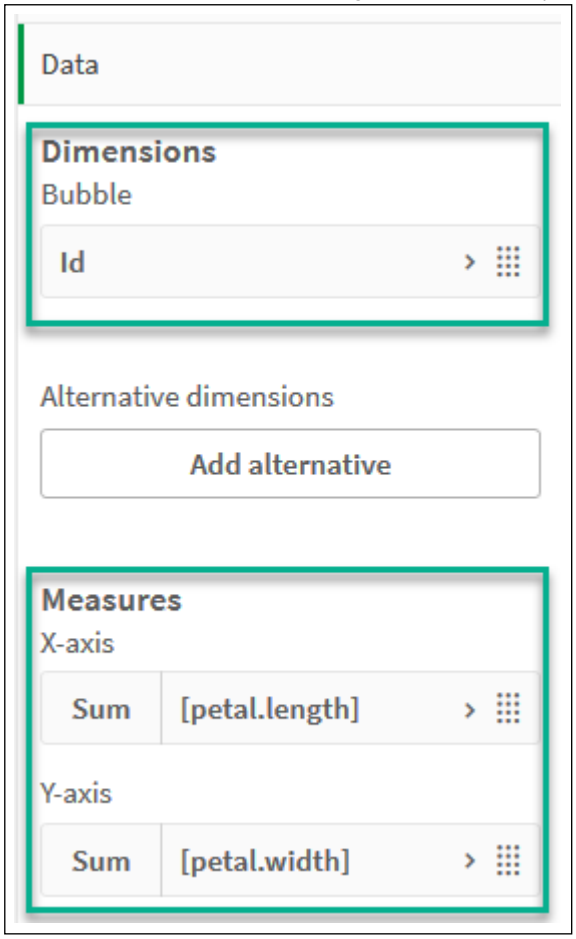
Nous créons également une variable pour l'argument *num_iter*, puis nous utilisons une deuxième zone d'entrée de variable pour modifier le nombre d'itérations.

L'ensemble de données *Iris* est publiquement disponible dans une variété de formats. Nous avons fourni les données sous forme de tableau intégré à charger via l'éditeur de chargement de données dans Qlik Sense. Notez que nous avons ajouté une colonne *Id* à la table de données pour cet exemple.

Après avoir chargé les données dans Qlik Sense, procédez comme suit :

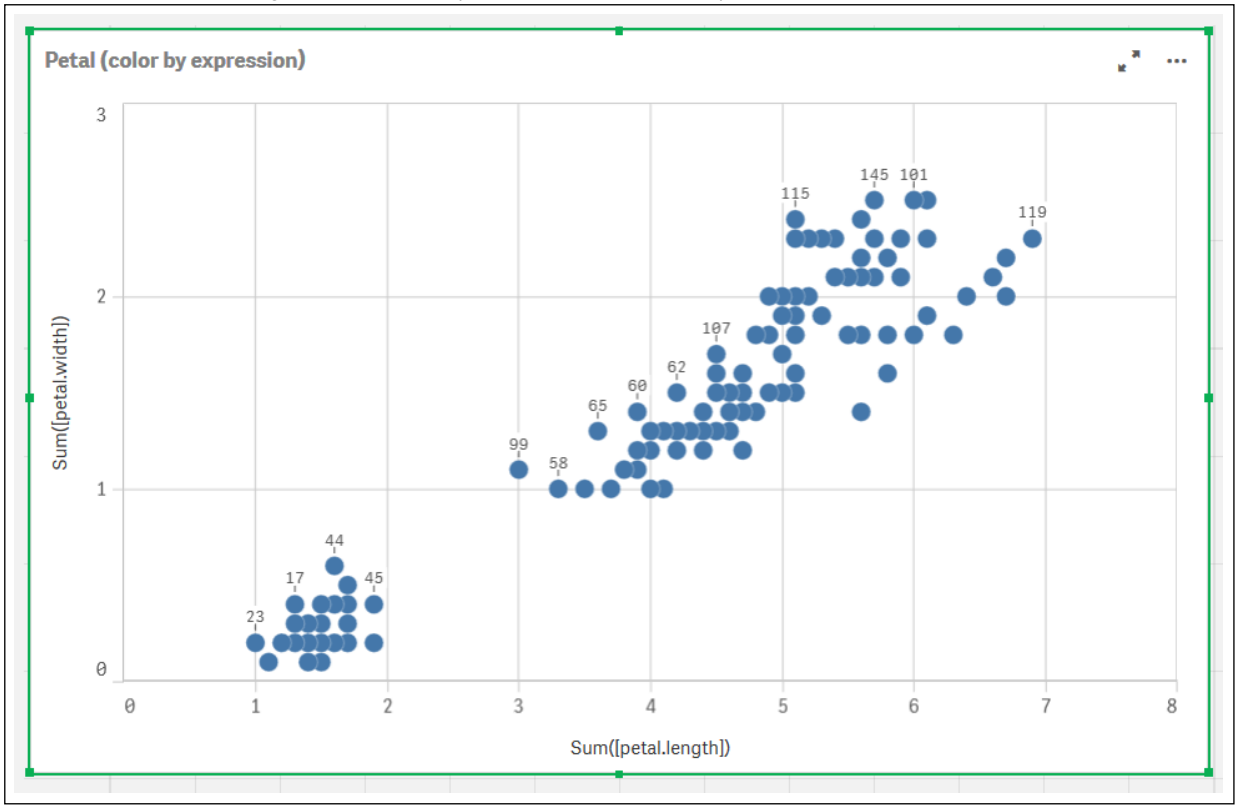
1. Glissez un graphique **Nuage de points** sur une nouvelle feuille. Nommez le graphique *Pétale (expression de la couleur)*.
2. Créez une variable pour spécifier le nombre de clusters. Pour la variable **Nom**, saisissez *KmeansPetalClusters*. Pour la variable **Définition**, saisissez *=2*.
3. Créez une variable pour spécifier le nombre d'itérations. Pour la variable **Nom**, saisissez *KmeansNumberIterations*. Pour la variable **Définition**, saisissez *=1*.
4. Configurez **Données** pour le graphique :
    i. sous **Dimensions**, sélectionnez *id* pour le champ **Bulle**. Saisissez l'Id de cluster de l'Étiquette.
    ii. Sous **Mesures**, sélectionnez *Sum([petal.length])* pour l'expression **Axe X**.
    iii. Sous **Mesures**, sélectionnez *Sum([petal.width])* pour l'expression **Axe Y**.

Paramètres des données pour le graphique Pétale (expression de la couleur)



Les points de données sont tracés sur le graphique.

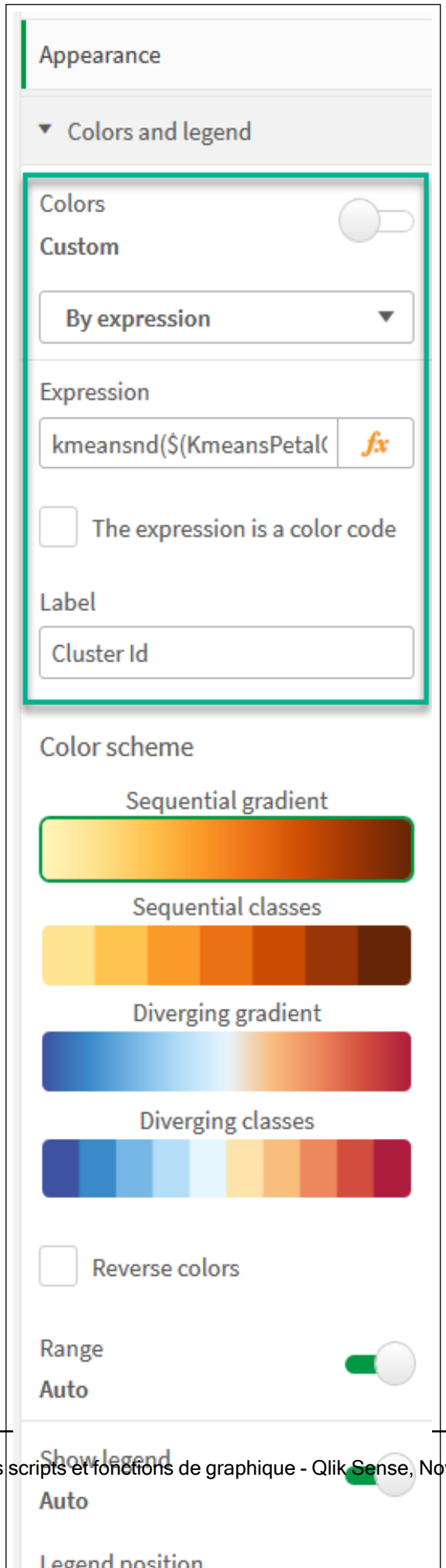
Points de données sur le graphique Pétale (expression de la couleur)



5. Configurez **Aspect** pour le graphique :
   i. Sous **Couleurs et légende**, sélectionnez **Personnaliser** pour **Couleurs**.
   ii. Choisissez de colorer le graphique **Par expression**.
   iii. Saisissez la valeur suivante pour **Expression** : *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   Notez que *KmeansPetalClusters* est la variable définie sur *2*. *KmeansNumberIterations* est la variable définie sur *1*.
   Sinon, saisissez la valeur suivante : *kmeansnd(2, 2, Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   iv. Décochez la case **Expression sous forme de code couleur**.

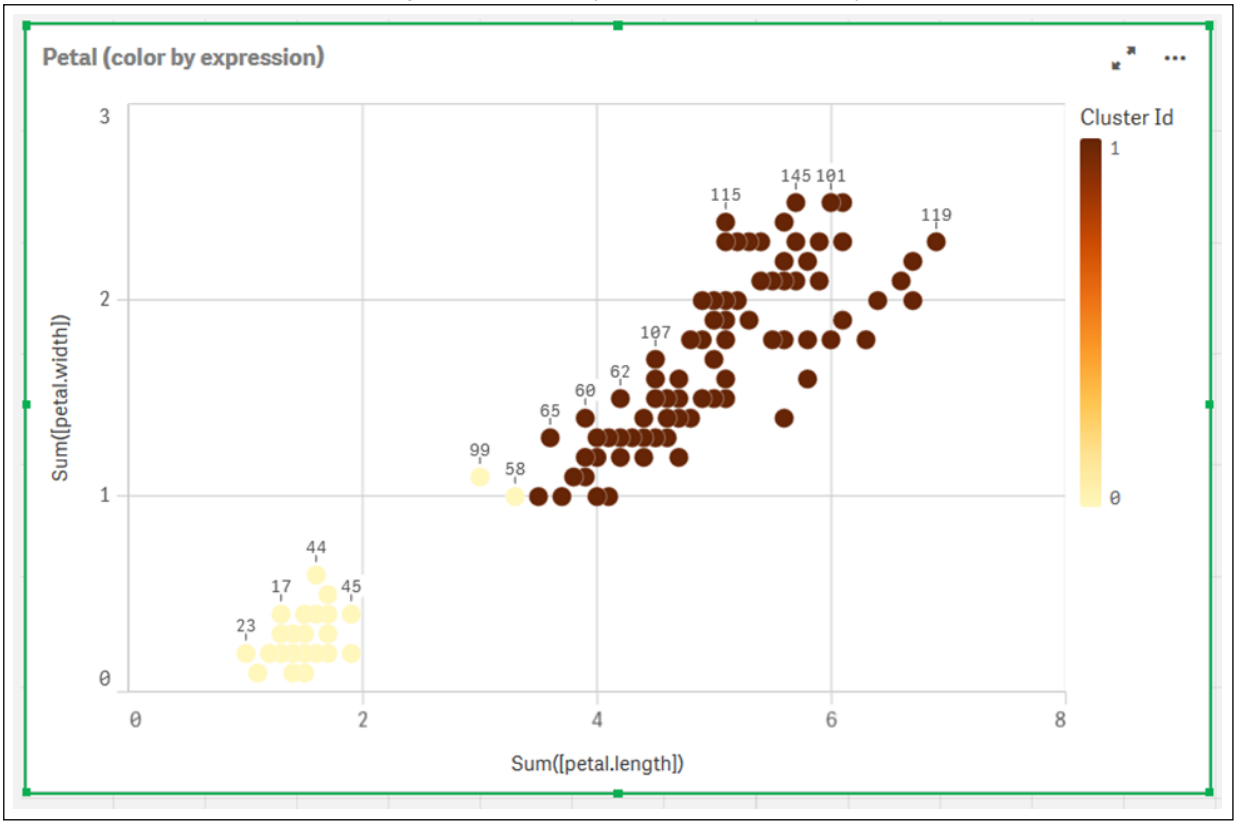
v. Saisissez la valeur suivante pour **Étiquette** : *Id de cluster*

Paramètres d'aspect pour le graphique Pétale (expression de la couleur)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position

Les deux clusters du graphique sont colorés par l'expression KMeans.

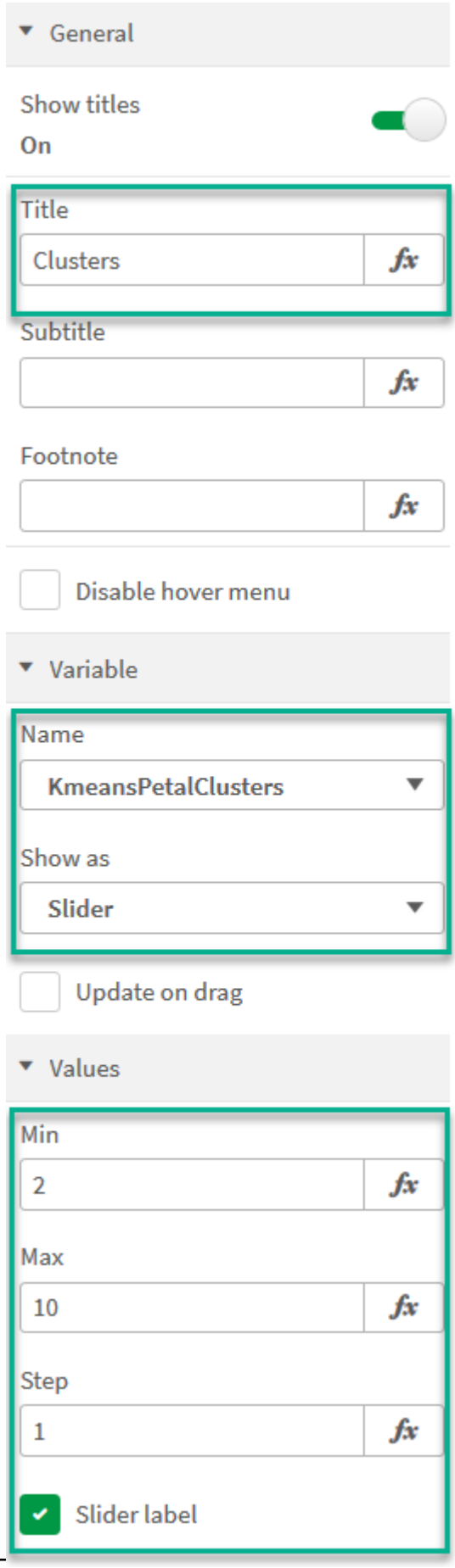
Clusters colorés par expression sur le graphique Pétale (expression de la couleur)



6. Ajoutez une zone **Entrée de variable** pour le nombre de clusters.
    i. Sous **Objets personnalisés** dans le panneau des **Ressources**, sélectionnez **Qlik Dashboard bundle**. Si nous n'avions pas accès au Dashboard bundle, nous pourrions tout de même modifier le nombre de clusters à l'aide de la variable créée, ou directement sous forme d'entier dans l'expression.
    ii. Glissez une zone **Entrée de variable** sur la feuille.
    iii. Sous **Aspect**, cliquez sur **Général**.
    iv. Saisissez la valeur suivante pour **Titre** : *Clusters*
    v. Cliquez sur **Variable**.
    vi. Sélectionnez la variable suivante pour **Nom** : *KmeansPetalClusters*.
    vii. Sélectionnez **Curseur** pour **Afficher comme**.

viii. Sélectionnez **Valeurs** et configurez les paramètres selon les besoins.

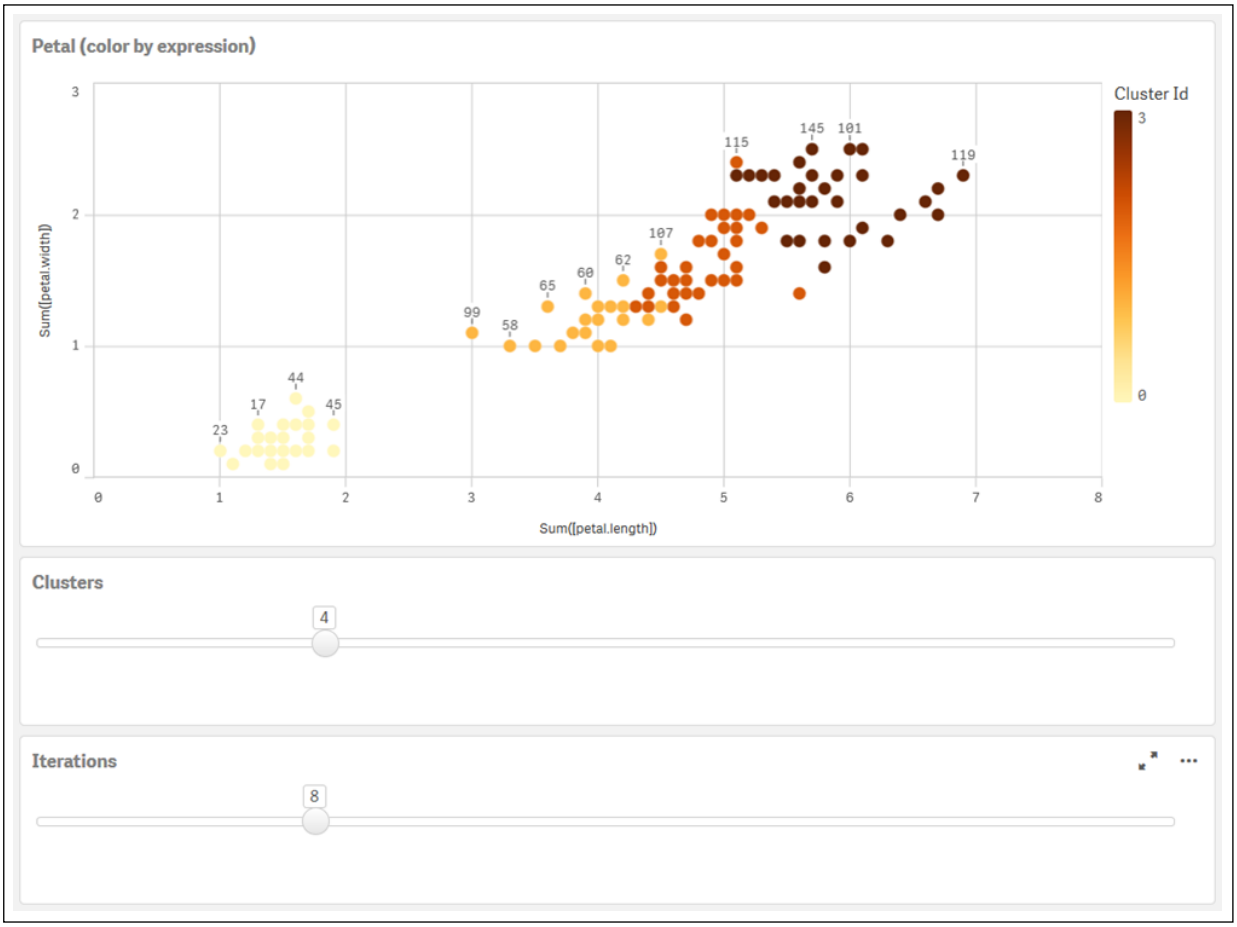
Aspect pour la zone d'entrée de variable Clusters


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

7. Ajoutez une zone **Entrée de variable** pour le nombre d'itérations.
    i. Glissez une zone **Entrée de variable** sur la feuille.
    ii. Sous **Aspect**, sélectionnez **Général**.
    iii. Saisissez la valeur suivante pour **Titre** : *Itérations*
    iv. Sous **Aspect**, sélectionnez **Variable**.
    v. Sélectionnez la variable suivante sous **Nom** : *KmeansNumberIterations*.
    vi. Configurez les paramètres supplémentaires selon les besoins.

    À présent, nous pouvons modifier le nombre de clusters et d'itérations à l'aide des curseurs des zones d'entrée de variable.

    Clusters colorés par expression sur le graphique Pétale (expression de la couleur)

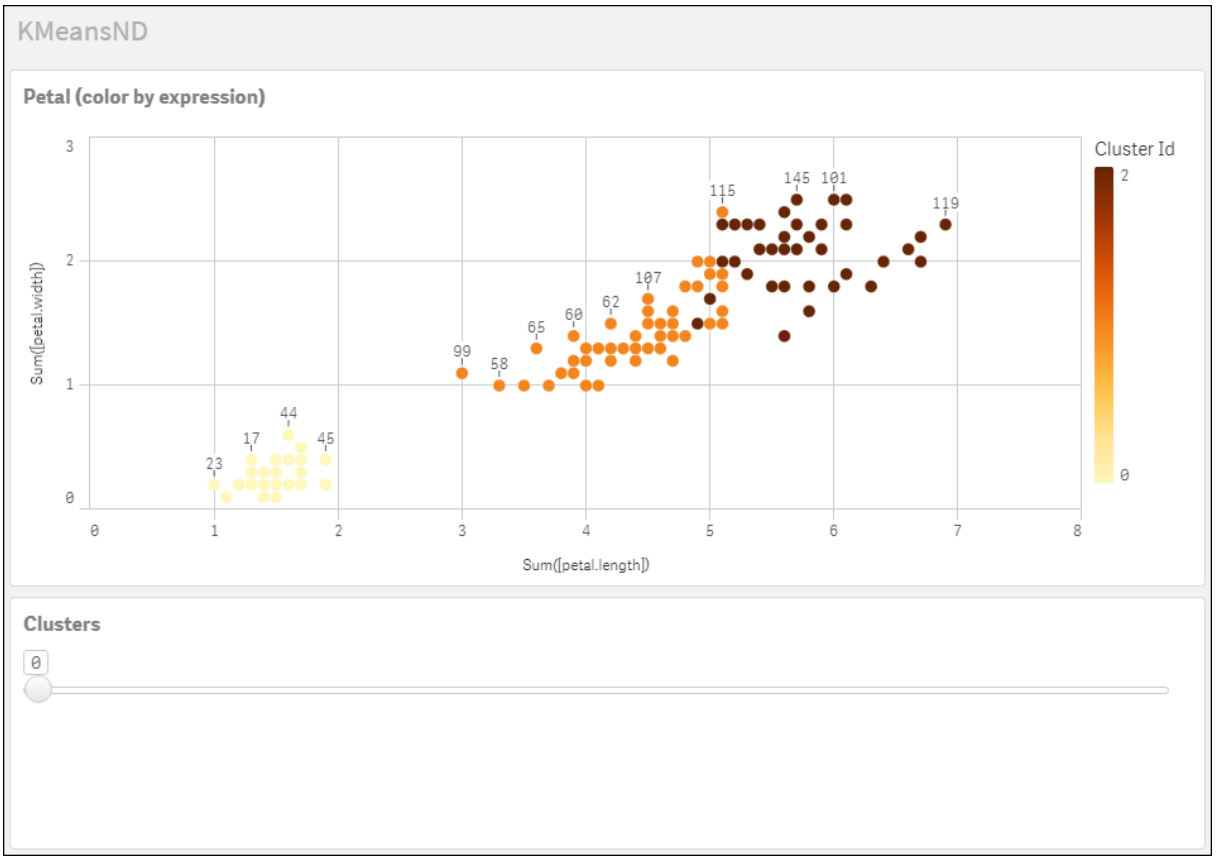


Clustering automatique

Les fonctions **K-moyennes** prennent en charge le clustering automatique via une méthode dite Différence de profondeur (DeD - Depth Difference). Quand un utilisateur définit 0 comme nombre de clusters, un nombre optimal de clusters est déterminé pour cet ensemble de données. Notez que même si entier n'est pas explicitement renvoyé pour le nombre de clusters (*k*), il est calculé dans l'algorithme K-moyennes. Par exemple, si 0 est spécifié dans la fonction pour la valeur de *KmeansPetalClusters* ou défini via une zone

d'entrée de variable, les affectations de clusters sont automatiquement calculées pour l'ensemble de données en fonction d'un nombre optimal de clusters. Étant donné l'ensemble de données Iris, si 0 est sélectionné comme nombre de clusters, l'algorithme déterminera un nombre optimal de clusters (3) pour cet ensemble de données (c'est ce qu'on appelle le clustering automatique).

La méthode Différence de profondeur de K-moyennes détermine le nombre optimal de clusters quand (k) est défini sur 0.



Ensemble de données Iris : Chargement intégré de l'éditeur de chargement de données dans Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
```

```
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansCentroid2D - fonction de graphique

KMeansCentroid2D() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il affiche la coordonnée souhaitée du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'agorithme sont déterminées par les paramètres coordinate_1 et coordinate_2, respectivement. Ces deux paramètres sont des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters. En option, les données peuvent être normalisées par le paramètre de norme.

KMeansCentroid2D renvoie une valeur par point de données. La valeur renvoyée est une valeur double et est l'une des coordonnées de la position correspondant au centre de cluster auquel chaque point de données a été affecté.

Syntaxe :

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
num_clusters	Entier qui spécifie le nombre de clusters.
coordinate_no	Le nombre de coordonnées souhaité des centroïdes (correspondant, par exemple, à l'axe x, y ou z).
coordinate_1	L'agrégation calcule la première coordonnée, généralement l'axe x du nuage de points qui peut être obtenu à partir du graphique. Le paramètre supplémentaire, coordinate_2, calcule la deuxième coordonnée.
norm	La méthode de normalisation optionnelle est appliquée aux ensembles de données avant le clustering KMeans. Valeurs possibles : 0 ou 'none' pour aucune normalisation 1 ou 'zscore' pour la normalisation z-score 2 ou 'minmax' pour la normalisation min-max Si aucun paramètre n'est fourni ou si le paramètre fourni est incorrect, aucune normalisation n'est appliquée. Z-score normalise les données en fonction d'une moyenne des fonctions et d'un écart-type standard. Z-score ne garantit pas que chaque fonction a la même échelle, mais il s'agit d'une meilleure approche que min-max pour traiter les valeurs hors norme. La normalisation min-max garantit que les fonctions ont la même échelle en prenant les valeurs minimale et maximale et chacune et en recalculant chaque point de données.

Clustering automatique

Les fonctions **K-moyennes** prennent en charge le clustering automatique via une méthode dite Différence de profondeur (DeD - Depth Difference). Quand un utilisateur définit 0 comme nombre de clusters, un nombre optimal de clusters est déterminé pour cet ensemble de données. Notez que même si entier n'est pas explicitement renvoyé pour le nombre de clusters (*k*), il est calculé dans l'algorithme K-moyennes. Par

exemple, si 0 est spécifié dans la fonction pour la valeur de *KmeansPetalClusters* ou défini via une zone d'entrée de variable, les affectations de clusters sont automatiquement calculées pour l'ensemble de données en fonction d'un nombre optimal de clusters.

KMeansCentroidND - fonction de graphique

KMeansCentroidND() évalue les lignes du graphique en appliquant un algorithme des k-moyennes, et, pour chaque ligne du graphique, il affiche la coordonnée souhaitée du cluster auquel ce point de données a été affecté. Les colonnes utilisées par l'algorithme sont déterminées par les paramètres coordinate_1, coordinate_2, etc., jusqu'à n colonnes. Ces paramètres sont tous des agrégations. Le nombre de clusters créés est déterminé par le paramètre num_clusters.

KMeansCentroidND renvoie une valeur par ligne. La valeur renvoyée est une valeur double et est l'une des coordonnées de la position correspondant au centre de cluster auquel chaque point de données a été affecté.

Syntaxe :

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Type de données renvoyé : double

Arguments :

Arguments

Argument	Description
num_clusters	Entier qui spécifie le nombre de clusters.
num_iter	Le nombre d'itérations de clustering avec des centres de cluster réinitialisés.
coordinate_no	Le nombre de coordonnées souhaité des centroïdes (correspondant, par exemple, à l'axe x, y ou z).
coordinate_1	L'agrégation calcule la première coordonnée, généralement l'axe x (d'un nuage de points qui peut être obtenu à partir du graphique). Les paramètres supplémentaires calculent les deuxième, troisième et quatrième coordonnées, etc.

Clustering automatique

Les fonctions **K-moyennes** prennent en charge le clustering automatique via une méthode dite Différence de profondeur (DeD - Depth Difference). Quand un utilisateur définit 0 comme nombre de clusters, un nombre optimal de clusters est déterminé pour cet ensemble de données. Notez que même si entier n'est pas explicitement renvoyé pour le nombre de clusters (*k*), il est calculé dans l'algorithme K-moyennes. Par exemple, si 0 est spécifié dans la fonction pour la valeur de *KmeansPetalClusters* ou défini via une zone d'entrée de variable, les affectations de clusters sont automatiquement calculées pour l'ensemble de données en fonction d'un nombre optimal de clusters.

5.23 Fonctions de distribution statistiques

Les fonctions de distribution statistiques renvoient les probabilités d'occurrence de différents résultats possibles d'une variable d'entrée donnée. Vous pouvez utiliser ces fonctions pour calculer les valeurs potentielles de vos points de données.

Les trois groupes de fonctions de distribution statistiques décrits ci-après sont tous implémentés dans Qlik Sense au moyen de la bibliothèque de fonctions Cephes. Pour obtenir des références et des informations détaillées sur les algorithmes utilisés, la précision, etc., consultez le site Web : ≤ Cephes library. La bibliothèque de fonctions Cephes est accessible par autorisation.

- Les fonctions de probabilité calculent la probabilité au point de la distribution donné par la valeur fournie.
    - Les fonctions Frequency sont utilisées pour les distributions discrètes.
    - Les fonctions Density sont utilisées pour les fonctions continues.
- Les fonctions Dist calculent la probabilité cumulée de la distribution au point de la distribution donné par la valeur fournie.
- Les fonctions Inv calculent la valeur inverse, suivant la probabilité cumulée de la distribution.

Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

Vue d'ensemble des fonctions de distribution statistiques

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

BetaDensity

`BetaDensity()` renvoie la probabilité de la distribution bêta.

```
BetaDensity (value, alpha, beta)
```

BetaDist

`BetaDist()` renvoie la probabilité cumulée de la distribution bêta.

```
BetaDist (value, alpha, beta)
```

BetaInv

`BetaINV()` renvoie l'inverse de la probabilité cumulée de la distribution bêta.

```
BetaInv (prob, alpha, beta)
```

BinomDist

`BinomDist()` renvoie la probabilité cumulée de la distribution binomiale.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency

`BinomFrequency()` renvoie la distribution de la probabilité binomiale.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` renvoie l'inverse de la probabilité cumulée de la distribution binomiale.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` renvoie la probabilité unilatérale de la distribution chi^2. La fonction de densité chi^2 est associée à un test chi^2.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` renvoie la probabilité unilatérale de la distribution chi^2. La distribution chi^2 est associée à un test chi^2.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` renvoie l'inverse de la probabilité unilatérale de la distribution de chi^2.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` renvoie la probabilité de la distribution `F`.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` renvoie la probabilité cumulée de la distribution `F`.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` renvoie l'inverse de la probabilité cumulée de la distribution `F`.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` renvoie la probabilité de la distribution gamma.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` renvoie la probabilité cumulée de la distribution gamma.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` renvoie l'inverse de la probabilité cumulée de la distribution gamma.

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` renvoie la distribution normale cumulative pour la moyenne et l'écart type spécifiés. Si mean = 0 et standard_dev = 1, la fonction renvoie la distribution normale type.

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` renvoie l'inverse de la distribution normale cumulative pour la moyenne et l'écart type spécifiés.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` renvoie la probabilité cumulée de la distribution Poisson.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` renvoie la distribution de la probabilité Poisson.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` renvoie l'inverse de la probabilité cumulée de la distribution Poisson.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` renvoie la valeur de la fonction de densité `t` de l'étudiant où une valeur numérique est une valeur calculée de `t` dont la probabilité doit être calculée.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` renvoie la probabilité pour la distribution `t` de l'étudiant où une valeur numérique est une valeur calculée de `t` dont la probabilité doit être calculée.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` renvoie la valeur `t` de la distribution `t` de l'étudiant sous forme de fonction de la probabilité et des degrés de liberté.

```
TInv (prob, degrees_freedom)
```

Voir aussi :

p *Fonctions d'agrégation statistiques (page 391)*

BetaDensity

`BetaDensity()` renvoie la probabilité de la distribution bêta.

Syntaxe :

```
BetaDensity(value, alpha, beta)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur doit être comprise entre 0 et 1.
`alpha`	Nombre positif définissant le premier paramètre de forme. Il s'agit de l'exposant de la variable aléatoire.
`beta`	Nombre positif définissant le deuxième paramètre de forme. Il indique le nombre de degrés de liberté du dénominateur.

BetaDist

`BetaDist()` renvoie la probabilité cumulée de la distribution bêta.

Syntaxe :

```
BetaDist(value, alpha, beta)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur doit être comprise entre 0 et 1.
`alpha`	Nombre positif définissant le premier paramètre de forme. Il s'agit de l'exposant de la variable aléatoire.
`beta`	Nombre positif définissant le deuxième paramètre de forme. Il s'agit de l'exposant qui contrôle la forme de la distribution.

Cette fonction est liée à la fonction `BetaInv` de la manière suivante :
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` renvoie l'inverse de la probabilité cumulée de la distribution bêta.

Syntaxe :

```
BetaInv(prob, alpha, beta)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
prob	Probabilité associée à la distribution de la probabilité bêta. Elle doit correspondre à un nombre compris entre 0 et 1.
alpha	Nombre positif définissant le premier paramètre de forme. Il s'agit de l'exposant de la variable aléatoire.
beta	Nombre positif définissant le deuxième paramètre de forme. Il s'agit de l'exposant qui contrôle la forme de la distribution.

Cette fonction est liée à la fonction `BetaDist` de la manière suivante :
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` renvoie la probabilité cumulée de la distribution binomiale.

Syntaxe :

```
BinomDist(value, trials, trial_probability)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur doit être un entier non inférieur à zéro et non supérieur au nombre d'essais.
trials	Entier positif indiquant le nombre d'essais.
trial_probability	Probabilité de réussite de chaque essai. Il s'agit toujours d'un nombre compris entre 0 et 1.

Cette fonction est liée à la fonction `BinomInv` de la manière suivante :
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` renvoie la distribution de la probabilité binomiale.

Syntaxe :

```
BinomFrequency(value, trials, trial_probability)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur doit être un entier non inférieur à zéro et non supérieur au nombre d'essais.
`trials`	Entier positif indiquant le nombre d'essais.
`trial_probability`	Probabilité de réussite de chaque essai. Il s'agit toujours d'un nombre compris entre 0 et 1.

BinomInv

`BinomInv()` renvoie l'inverse de la probabilité cumulée de la distribution binomiale.

Syntaxe :

```
BinomInv(prob, trials, trial_probability)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`prob`	Probabilité associée à la distribution de la probabilité binomiale. Elle doit correspondre à un nombre compris entre 0 et 1.
`trials`	Entier positif indiquant le nombre d'essais.
`trial_probability`	Probabilité de réussite de chaque essai. Il s'agit toujours d'un nombre compris entre 0 et 1.

Cette fonction est liée à la fonction `BinomDist` de la manière suivante :
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

`ChiDensity()` renvoie la probabilité unilatérale de la distribution chi^2. La fonction de densité chi^2 est associée à un test chi^2.

Syntaxe :

```
ChiDensity(value, degrees_freedom)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
degrees_ freedom	Entier positif indiquant le nombre de degrés de liberté du numérateur.

ChiDist

ChiDist() renvoie la probabilité unilatérale de la distribution chi^2. La distribution chi^2 est associée à un test chi^2.

Syntaxe :

```
CHIDIST(value, degrees_freedom)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
degrees_ freedom	Entier positif indiquant le nombre de degrés de liberté.

Cette fonction est liée à la fonction **ChiInv** de la manière suivante :
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
CHIDIST(8, 15)	Renvoie 0.9238.

ChiInv

ChiInv() renvoie l'inverse de la probabilité unilatérale de la distribution de chi^2.

Syntaxe :

```
CHIINV(prob, degrees_freedom)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
prob	Probabilité associée à la distribution chi^2. Elle doit correspondre à un nombre compris entre 0 et 1.
degrees_ freedom	Entier indiquant le nombre de degrés de liberté.

Cette fonction est liée à la fonction **ChiDist** de la manière suivante :
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
CHIINV(0.9237827, 15)	Renvoie 8.0000.

FDensity

`FDensity()` renvoie la probabilité de la distribution `F`.

Syntaxe :

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
`degrees_ freedom1`	Entier positif indiquant le nombre de degrés de liberté du numérateur.
`degrees_ freedom2`	Entier positif indiquant le nombre de degrés de liberté du dénominateur.

FDist

`FDist()` renvoie la probabilité cumulée de la distribution `F`.

Syntaxe :

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
degrees_ freedom1	Entier positif indiquant le nombre de degrés de liberté du numérateur.
degrees_ freedom2	Entier positif indiquant le nombre de degrés de liberté du dénominateur.

Cette fonction est liée à la fonction **FInv** de la manière suivante :
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
FDIST(15, 8, 6)	Renvoie 0.0019.

FInv

`FInv()` renvoie l'inverse de la probabilité cumulée de la distribution `F`.

Syntaxe :

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
prob	Probabilité associée à la distribution de probabilité F et devant correspondre à une valeur comprise entre 0 et 1.
degrees_ freedom	Entier indiquant le nombre de degrés de liberté.

Cette fonction est liée à la fonction **FDist** de la manière suivante :
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
FINV(0.0019369, 8, 6)	Renvoie 15.0000.

GammaDensity

`GammaDensity()` renvoie la probabilité de la distribution gamma.

Syntaxe :

```
GammaDensity(value, k, θ)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
`k`	Nombre positif définissant le paramètre de forme.
`θ`	Nombre positif définissant le paramètre d'échelle.

GammaDist

`GammaDist()` renvoie la probabilité cumulée de la distribution gamma.

Syntaxe :

```
GammaDist(value, k, θ)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
`k`	Nombre positif définissant le paramètre de forme.
`θ`	Nombre positif définissant le paramètre d'échelle.

Cette fonction est liée à la fonction GammaINv de la manière suivante :
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` renvoie l'inverse de la probabilité cumulée de la distribution gamma.

Syntaxe :

```
GammaInv(prob, k, θ)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
prob	Probabilité associée à la distribution de la probabilité gamma. Elle doit correspondre à un nombre compris entre 0 et 1.
k	Nombre positif définissant le paramètre de forme.
θ	Nombre positif définissant le paramètre d'échelle.

Cette fonction est liée à la fonction GammaDist de la manière suivante :
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` renvoie la distribution normale cumulative pour la moyenne et l'écart type spécifiés. Si mean = 0 et standard_dev = 1, la fonction renvoie la distribution normale type.

Syntaxe :

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution.
mean	Valeur facultative indiquant la moyenne arithmétique de la distribution. Si vous ne précisez pas cet argument, la valeur par défaut est 0.
standard_ dev	Valeur positive facultative indiquant l'écart type de la distribution. Si vous ne précisez pas cet argument, la valeur par défaut est 1.

Argument	Description
cumulative	Si vous le souhaitez, vous pouvez choisir d'utiliser une distribution normale type ou une distribution cumulative. 0 = distribution normale type 1 = distribution cumulative (valeur par défaut)

Cette fonction est liée à la fonction **NormInv** de la manière suivante :
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
NORMDIST(0.5, 0, 1)	Renvoie 0.6915.

NormInv

`NormInv()` renvoie l'inverse de la distribution normale cumulative pour la moyenne et l'écart type spécifiés.

Syntaxe :

```
NORMINV(prob, mean, standard_dev)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
prob	Probabilité associée à la distribution normale. Elle doit correspondre à un nombre compris entre 0 et 1.
mean	Valeur indiquant la moyenne arithmétique de la distribution.
standard_ dev	Valeur positive indiquant l'écart type de la distribution.

Cette fonction est liée à la fonction **NormDist** de la manière suivante :
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
NORMINV(0.6914625, 0, 1)	Renvoie 0.5000.

PoissonDist

`PoissonDist()` renvoie la probabilité cumulée de la distribution Poisson.

Syntaxe :

```
PoissonDist(value, mean)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
`mean`	Nombre positif définissant le résultat moyen.

Cette fonction est liée à la fonction `PoissonInv` de la manière suivante :
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` renvoie la distribution de la probabilité Poisson.

Syntaxe :

```
PoissonFrequency(value, mean)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
`value`	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
`mean`	Nombre positif définissant le résultat moyen.

PoissonInv

`PoissonInv()` renvoie l'inverse de la probabilité cumulée de la distribution Poisson.

Syntaxe :

```
PoissonInv(prob, mean)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
prob	Probabilité associée à la distribution de la probabilité Poisson. Elle doit correspondre à un nombre compris entre 0 et 1.
mean	Nombre positif définissant le résultat moyen.

Cette fonction est liée à la fonction PoissonDIST de la manière suivante :
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` renvoie la valeur de la fonction de densité `t` de l'étudiant où une valeur numérique est une valeur calculée de `t` dont la probabilité doit être calculée.

Syntaxe :

```
TDensity(value, degrees_freedom)
```

Type de données renvoyé : nombre

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
degrees_ freedom	Entier positif indiquant le nombre de degrés de liberté.

TDist

`TDist()` renvoie la probabilité pour la distribution `t` de l'étudiant où une valeur numérique est une valeur calculée de `t` dont la probabilité doit être calculée.

Syntaxe :

```
TDist(value, degrees_freedom, tails)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
value	Valeur à laquelle vous souhaitez évaluer la distribution. La valeur ne doit pas être négative.
degrees_ freedom	Entier positif indiquant le nombre de degrés de liberté.
tails	Doit être égal à 1 (distribution unilatérale) ou à 2 (distribution bilatérale).

Cette fonction est liée à la fonction **TInv** de la manière suivante :
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Exemples et résultats :

Exemple	Résultat
TDIST(1, 30, 2)	Renvoie 0.3253.

TInv

`TInv()` renvoie la valeur `t` de la distribution `t` de l'étudiant sous forme de fonction de la probabilité et des degrés de liberté.

Syntaxe :

```
TINV(prob, degrees_freedom)
```

Type de données renvoyé : nombre

Arguments :

Arguments

Argument	Description
prob	Probabilité bilatérale associée à la distribution t. Elle doit correspondre à un nombre compris entre 0 et 1.
degrees_ freedom	Entier indiquant le nombre de degrés de liberté.

Limitations :

Tous les arguments doivent être numériques, sinon la fonction renvoie la valeur NULL.

Cette fonction est liée à la fonction **TDist** de la manière suivante :
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Exemples et résultats :

Exemple	Résultat
TINV(0.3253086, 30)	Renvoie 1.0000.

5.24 Fonctions de chaîne

Cette section décrit les fonctions de gestion et de manipulation des chaînes.

Toutes les fonctions s'utilisent aussi bien dans le script de chargement de données que dans les expressions de graphique, à l'exception de la fonction **Evaluate** qui n'est admise que dans le script de chargement de données.

Vue d'ensemble des fonctions de chaîne

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Capitalize
Capitalize() renvoie la chaîne en affichant tous les mots en lettres majuscules.

```
Capitalize (text)
```

Chr
Chr() renvoie le caractère Unicode correspondant à l'entier d'entrée.

```
Chr (int)
```

Evaluate
Evaluate() détermine si la chaîne de texte d'entrée peut être évaluée en tant qu'expression Qlik Sense valide et, si tel est le cas, renvoie la valeur de l'expression sous forme de chaîne. Si la chaîne d'entrée n'est pas une expression valide, la valeur NULL est renvoyée.

```
Evaluate (expression_text)
```

FindOneOf
FindOneOf() recherche dans une chaîne la position de l'occurrence de n'importe quel caractère faisant partie d'un jeu de caractères fourni. La position de la première occurrence de n'importe quel caractère du jeu de recherche est renvoyée à moins qu'un troisième argument (doté d'une valeur supérieure à 1) ne soit fourni. En l'absence de correspondance, la valeur **0** est renvoyée.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() renvoie un hachage de 128 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 22 caractères.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() renvoie un hachage de 160 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 27 caractères.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() renvoie un hachage de 256 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 43 caractères.

```
Hash256 (expr{, expression})
```

Index

Index() recherche dans une chaîne la position de départ de la nième occurrence d'une sous-chaîne fournie. Un troisième argument facultatif fournit la valeur de n, qui est égale à 1 s'il est omis. Une valeur négative permet de lancer la recherche en commençant par la fin de la chaîne. Les positions dans la chaîne sont numérotées à partir de **1**.

```
Index  (text, substring[, count])
```

IsJson

IsJson() teste si une chaîne spécifiée contient des données JSON (JavaScript Object Notation) valides. Vous pouvez également valider un type de données JSON spécifique.

```
IsJson (json [, type])
```

JsonGet

JsonGet() renvoie le chemin d'accès à une chaîne de données JSON (JavaScript Object Notation). Les données doivent être conformes au format JSON, mais elles peuvent contenir des espaces supplémentaires ou de nouvelles lignes.

```
JsonGet (json, path)
```

JsonSet

JsonSet() modifie une chaîne contenant des données JSON (JavaScript Object Notation). Cette instruction peut définir ou insérer une valeur JSON avec le nouvel emplacement spécifié par le chemin d'accès. Les données doivent être conformes au format JSON, mais elles peuvent contenir des espaces supplémentaires ou de nouvelles lignes.

```
JsonSet(json, path, value)
```

KeepChar

KeepChar() renvoie une chaîne composée de la première chaîne ('text'), déduction faite des caractères NON contenus dans la deuxième chaîne ("keep_chars").

```
KeepChar (text, keep_chars)
```

Left

Left() renvoie une chaîne composée des premiers caractères de la chaîne d'entrée (en partant de la gauche) et dont le nombre de caractères est déterminé par le deuxième argument.

```
Left (text, count)
```

Len

Len() renvoie la longueur de la chaîne d'entrée.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() renvoie la distance Levenshtein entre deux chaînes. Cela est défini comme le nombre minimal d'éditions (insertions, suppression ou substitutions) d'un seul caractère requises pour modifier une chaîne dans l'autre. La fonction s'avère utile pour les comparaisons de chaînes partielles.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() convertit tous les caractères de la chaîne d'entrée en lettres minuscules.

```
Lower (text)
```

LTrim

LTrim() renvoie la chaîne d'entrée exempte de tout espace de début.

```
LTrim (text)
```

Mid

Mid() renvoie la partie de la chaîne d'entrée commençant à la position du caractère défini par le deuxième argument, « start », et renvoyant le nombre de caractères spécifié par le troisième argument, « count ». Si « count » est omis, c'est le reste de la chaîne d'entrée qui est renvoyé. Le premier caractère indiqué dans la chaîne d'entrée porte le numéro 1.

```
Mid (text, start[, count])
```

Ord

Ord() renvoie le numéro de point de code Unicode du premier caractère de la chaîne d'entrée.

```
Ord (text)
```

PurgeChar

PurgeChar() renvoie une chaîne composée des caractères contenus dans la chaîne d'entrée ('text'), à l'exception des caractères inclus dans le deuxième argument ('remove_chars').

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() forme une chaîne composée de la chaîne d'entrée répétée autant de fois que le nombre défini par le deuxième argument.

```
Repeat (text[, repeat_count])
```

Replace

Replace() renvoie une chaîne après avoir remplacé toutes les occurrences d'une sous-chaîne donnée dans la chaîne d'entrée par une autre sous-chaîne. La fonction n'est pas récursive et fonctionne de gauche à droite.

```
Replace (text, from_str, to_str)
```

Right

Right() renvoie une chaîne composée des derniers caractères (situés à l'extrémité droite) de la chaîne d'entrée et dont le nombre de caractères est déterminé par le deuxième argument.

```
Right (text, count)
```

RTrim

RTrim() renvoie la chaîne d'entrée exempte de tout espace de fin.

```
RTrim (text)
```

SubField

SubField() permet d'extraire des composants de sous-chaîne d'un champ de chaîne parent, où les champs d'enregistrement d'origine se composent de plusieurs parties séparées par un délimiteur.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() renvoie le nombre d'occurrences de la sous-chaîne spécifiée dans le texte de la chaîne d'entrée. Si aucun caractère ne correspond, la fonction renvoie 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() renvoie le texte de la chaîne d'entrée figurant entre les caractères spécifiés comme délimiteurs.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() renvoie la chaîne d'entrée exempte de tout espace de début et de fin.

```
Trim (text)
```

Upper

Upper() convertit les caractères de la chaîne d'entrée en lettres majuscules pour tous les caractères de texte de l'expression. Les nombres et les symboles sont ignorés.

```
Upper (text)
```

Capitalize

Capitalize() renvoie la chaîne en affichant tous les mots en lettres majuscules.

Syntaxe :

```
Capitalize(text)
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
`Capitalize ( 'star trek' )`	Renvoie 'Star Trek'.
`Capitalize ( 'AA bb cC Dd')`	Renvoie 'Aa Bb Cc Dd'.

Exemple : Script de chargement

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Résultat

Chaîne	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() renvoie le caractère Unicode correspondant à l'entier d'entrée.

Syntaxe :

```
Chr(int)
```

Type de données renvoyé : chaîne

Exemples et résultats :

Exemple	Résultat
Chr(65)	Renvoie la chaîne 'A'.
Chr(163)	Renvoie la chaîne '£'.
Chr(35)	Renvoie la chaîne '#'.

Evaluate

Evaluate() détermine si la chaîne de texte d'entrée peut être évaluée en tant qu'expression Qlik Sense valide et, si tel est le cas, renvoie la valeur de l'expression sous forme de chaîne. Si la chaîne d'entrée n'est pas une expression valide, la valeur NULL est renvoyée.

Syntaxe :

```
Evaluate(expression_text)
```

Type de données renvoyé : double

Cette fonction de chaîne ne s'utilise pas dans les expressions de graphique.

Exemples et résultats :

Exemple de fonction	Résultat
`Evaluate ( 5 * 8 )`	Renvoie '40'.

Exemple de script de chargement

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Résultat

Chaîne	Évalué
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() recherche dans une chaîne la position de l'occurrence de n'importe quel caractère faisant partie d'un jeu de caractères fourni. La position de la première occurrence de n'importe quel caractère du jeu de recherche est renvoyée à moins qu'un troisième argument (doté d'une valeur supérieure à 1) ne soit fourni. En l'absence de correspondance, la valeur **0** est renvoyée.

Syntaxe :

```
FindOneOf(text, char_set[, count])
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
char_set	Jeu de caractères à rechercher dans text.
count	Définit l'occurrence d'un caractère à rechercher. Par exemple, une valeur de 2 recherche la deuxième occurrence.

Exemple : Expressions de graphique

Exemple	Résultat
`FindOneOf( 'my example text string', 'et%s')`	Renvoie '4', car ‘e’ est le quatrième caractère de l'exemple de chaîne.
`FindOneOf( 'my example text string', 'et%s', 3)`	Renvoie '12', car la recherche porte sur n'importe lequel des caractères e, t, % ou s, et "t" est la troisième occurrence à la position 12 de l'exemple de chaîne.
`FindOneOf( 'my example text string', '¤%&')`	Renvoie '0', car aucun des caractères ¤, % ou & n'existe dans l'exemple de chaîne.

Exemple : Script de chargement

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Résultat

SearchFor	Occurrence	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() renvoie un hachage de 128 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 22 caractères.

Syntaxe :

```
Hash128(expr{, expression})
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
Hash128 ('abc', 'xyz', '123')	Renvoie 'MA&5]6+3=:>:>G%S<U*S2+'.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Renvoie 'G7*=6GKPJ(Z+)^KM?<$'A+'.

Exemple : Script de chargement

```
Hash_128: Load *,  Hash128(Region, Year, Month) as Hash128; Load * inline [ Region, Year,
Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Résultat

Région	Année	Mois	Hash128
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() renvoie un hachage de 160 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 27 caractères.

Syntaxe :

```
Hash160(expr{, expression})
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
Hash160 ('abc', 'xyz', '123')	Renvoie 'MA&5]6+3=:>;>G%S<U*S2I:`=X*'.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Renvoie 'G7*=6GKPJ (Z+)^KM?<$'AI.)?U$'.

Exemple : Script de chargement

```
Hash_160: Load *,  Hash160(Region, Year, Month) as Hash160; Load * inline [ Region, Year,
Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Résultat

Région	Année	Mois	Hash160
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

Hash256

Hash256() renvoie un hachage de 256 bits des valeurs de l'expression d'entrée combinées. Le résultat est une chaîne de 43 caractères.

Syntaxe :

```
Hash256(expr{, expression})
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
Hash256 ('abc', 'xyz', '123')	Renvoie 'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ'.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Renvoie 'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:'.

Exemple : Script de chargement

```
Hash_256: Load *,  Hash256(Region, Year, Month) as Hash256; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Résultat

Région	Année	Mois	Hash256
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

Index

Index() recherche dans une chaîne la position de départ de la nième occurrence d'une sous-chaîne fournie. Un troisième argument facultatif fournit la valeur de n, qui est égale à 1 s'il est omis. Une valeur négative permet de lancer la recherche en commençant par la fin de la chaîne. Les positions dans la chaîne sont numérotées à partir de **1**.

Syntaxe :

```
Index(text, substring[, count])
```

Type de données renvoyé : entier

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
substring	Chaîne de caractères à rechercher dans text.
count	Définit l'occurrence de la sous-chaîne **substring** à rechercher. Par exemple, une valeur de 2 recherche la deuxième occurrence.

Exemples et résultats :

Exemple	Résultat
Index('abcdefg', 'cd')	Renvoie 3.
Index('abcdabcd', 'b', 2)	Renvoie 6 (la deuxième occurrence de 'b').
Index('abcdabcd', 'b',-2)	Renvoie 2 (la deuxième occurrence de 'b' en commençant par la fin).
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Renvoie 1997.
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Renvoie 07.

Exemple : Script

```
T1: Load  *,  index(String, 'cd') as Index_CD,              // returns 3  in Index_CD index
(String, 'b') as Index_B,               // returns 2 in Index_B index(String, 'b', -1) as
Index_B2;          // returns 2 or 6 in Index_B2 Load * inline [ String abcdefg abcdabcd ];
```

IsJson

IsJson() teste si une chaîne spécifiée contient des données JSON (JavaScript Object Notation) valides. Vous pouvez également valider un type de données JSON spécifique.

Syntaxe :

```
value IsJson(json [, type])
```

Type de données renvoyé : double

Arguments

Argument	Description
json	Chaîne à tester. Elle peut contenir des espaces supplémentaires ou de nouvelles lignes.
type	Argument facultatif qui spécifie le type de données JSON à tester. • 'value' (par défaut) • 'object' • 'array' • 'string' • 'number' • 'boolean' • 'null'

Exemple : Format JSON et type valides

Exemple	Résultat
`IsJson('null')`	Renvoie -1 (true).
`IsJson('"abc"', 'value')`	Renvoie -1 (true).
`IsJson('"abc"', 'string')`	Renvoie -1 (true).
`IsJson(123, 'number')`	Renvoie -1 (true).

Exemple : Format JSON ou type non valide

Exemple	Résultat	Description
`IsJson('text')`	Renvoie 0 (false).	'text' n'est pas une valeur JSON valide.
`IsJson('"text"', 'number')`	Renvoie 0 (false).	'"text"' n'est pas un nombre JSON valide.
`IsJson('"text"', 'text')`	Renvoie 0 (false).	'text' n'est pas un type JSON valide.

JsonGet

JsonGet() renvoie le chemin d'accès à une chaîne de données JSON (JavaScript Object Notation). Les données doivent être conformes au format JSON, mais elles peuvent contenir des espaces supplémentaires ou de nouvelles lignes.

Syntaxe :

```
value JsonGet(json, path)
```

Type de données renvoyé : double

Arguments

Argument	Description
json	Chaîne contenant des données JSON.
path	Le chemin d'accès doit être spécifié conformément à ≤ RFC 6901. Cela permettra la recherche de propriétés au sein des données JSON sans utiliser de fonctions de sous-chaîne ou d'index complexes.

Exemple : Format JSON et chemin d'accès valides

Exemple	Résultat
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Renvoie '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Renvoie '{"foo":"bar"}'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Renvoie '"bar"'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Renvoie '[123,"abc","ABC"]'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Renvoie '123'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Renvoie '"abc"'.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Renvoie '"ABC"'.

Exemple : Format JSON ou chemin d'accès non valide

Exemple	Résultat	Description
`JsonGet('{"a":"b"}','/b')`	Renvoie null.	Le chemin d'accès ne pointe pas vers une partie valide des données JSON.
`JsonGet('{"a"}','/a')`	Renvoie null.	Les données JSON ne sont pas au format JSON valide (le membre "a" n'a pas de valeur).

JsonSet

JsonSet() modifie une chaîne contenant des données JSON (JavaScript Object Notation). Cette instruction peut définir ou insérer une valeur JSON avec le nouvel emplacement spécifié par le chemin d'accès. Les données doivent être conformes au format JSON, mais elles peuvent contenir des espaces supplémentaires ou de nouvelles lignes.

Syntaxe :

```
value JsonSet(json, path, value)
```

Type de données renvoyé : double

Arguments

Argument	Description
json	Chaîne contenant des données JSON.
path	Le chemin d'accès doit être spécifié conformément à ≤ RFC 6901. Cela permet d'accumuler des propriétés au sein des données JSON sans utiliser de fonctions de sous-chaîne ou d'index complexes ni de concaténation.
value	Nouvelle valeur de chaîne au format JSON.

Exemple : Format JSON, chemin d'accès et valeur valides

Exemple	Résultat
`JsonSet('{}','/a','"b"')`	Renvoie '{"a":"b"}'.
`JsonSet('[]','/0','"x"')`	Renvoie '["x"]'.
`JsonSet('"abc"','','123')`	Renvoie 123.

Exemple : Format JSON, chemin d'accès ou valeur non valide

Exemple	Résultat	Description
`JsonSet ('"abc"','/x','123')`	Renvoie null.	Le chemin d'accès ne pointe pas vers une partie valide des données JSON.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Renvoie null.	Le chemin d'accès n'est pas valide.
`JsonSet(' {"a":"b"}','/a','abc')`	Renvoie null.	La valeur n'est pas au format JSON valide. La chaîne doit être encadrée par des guillemets.

KeepChar

KeepChar() renvoie une chaîne composée de la première chaîne ('text'), déduction faite des caractères NON contenus dans la deuxième chaîne ("keep_chars").

Syntaxe :

```
KeepChar(text, keep_chars)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
keep_chars	Chaîne contenant les caractères figurant dans text à conserver.

Exemple : Expressions de graphique

Exemple	Résultat
`KeepChar ( 'a1b2c3','123' )`	Renvoie 123.
`KeepChar ( 'a1b2c3','1234' )`	Renvoie 123.
`KeepChar ( 'a1b22c3','1234' )`	Renvoie 1223.
`KeepChar ( 'a1b2c3','312' )`	Renvoie 123.

Exemple : Script de chargement

```
T1: Load  *,  keepchar(String1, String2) as KeepChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *KeepChar* dans le script de chargement.

String1	String2	KeepChar
a1b2c3	123	123

Voir aussi :

p *PurgeChar (page 1452)*

Left

Left() renvoie une chaîne composée des premiers caractères de la chaîne d'entrée (en partant de la gauche) et dont le nombre de caractères est déterminé par le deuxième argument.

Syntaxe :

```
Left(text, count)
```

Type de données renvoyé : chaîne

Arguments :

Argument	Description
text	Chaîne d'origine.
count	Définit le nombre de caractères à inclure en partant de l'extrémité gauche de la chaîne **text**.

Exemple : Expression de graphique

Exemple	Résultat
`Left('abcdef', 3)`	Renvoie 'abc'.

Exemple : Script de chargement

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Résultat

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *Left* dans le script de chargement.

Texte	Début	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

p Voir également *Index (page 1442)*, qui permet une analyse de chaîne plus complexe.

Len

Len() renvoie la longueur de la chaîne d'entrée.

Syntaxe :

```
Len(text)
```

Type de données renvoyé : entier

Exemple : Expression de graphique

Exemple	Résultat
`Len('Peter')`	Renvoie 5.

Exemple : Script de chargement

```
T1: Load String, First&Second as NewString; Load *, mid(String,len(First)+1) as Second; Load *, upper(left(String,1)) as First; Load  * inline [ String this is a sample text string capitalize first letter only ];
```

Résultat

Chaîne	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() renvoie la distance Levenshtein entre deux chaînes. Cela est défini comme le nombre minimal d'éditions (insertions, suppression ou substitutions) d'un seul caractère requises pour modifier une chaîne dans l'autre. La fonction s'avère utile pour les comparaisons de chaînes partielles.

Syntaxe :

```
LevenshteinDist(text1, text2)
```

Type de données renvoyé : entier

Exemple : Expression de graphique

Exemple	Résultat
`LevenshteinDist('Kitten','Sitting')`	Renvoie '3'.

Exemple : Script de chargement

Script de chargement

```
T1: Load *, recno() as ID; Load 'Silver' as String_1,* inline [ String_2 Sliver SSiver SSiveer ];  T1: Load *, recno()+3 as ID; Load 'Gold' as String_1,* inline [ String_2 Bold Bool Bond ];  T1: Load *, recno()+6 as ID; Load 'Ove' as String_1,* inline [ String_2 Ove Uve Üve ];  T1:
```

```
Load *, recno()+9 as ID; Load 'ABC' as String_1,* inline [ String_2 DEFG abc ビビビ ];  set
nullinterpret = '<NULL>'; T1: Load *, recno()+12 as ID; Load 'X' as String_1,* inline [
String_2 '' <NULL> 1 ];  R1: Load  ID,  String_1,  String_2,  LevenshteinDist(String_1,
String_2) as LevenshteinDistance  resident T1;  Drop table T1;
```

Résultat

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Gras	1
5	Gold	Bool	3
6	Gold	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1
10	ABC	DEFG	4
11	ABC	abc	3
12	ABC	ビビビ	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() convertit tous les caractères de la chaîne d'entrée en lettres minuscules.

Syntaxe :

```
Lower(text)
```

Type de données renvoyé : chaîne

Exemple : Expression de graphique

Exemple	Résultat
`Lower('abcD')`	Renvoie 'abcd'.

Exemple : Script de chargement

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Résultat

Chaîne	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() renvoie la chaîne d'entrée exempte de tout espace de début.

Syntaxe :

```
LTrim(text)
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
`LTrim( ' abc' )`	Renvoie 'abc'.
`LTrim( 'abc ' )`	Renvoie 'abc '.

Exemple : Script de chargement

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

> *L'instruction "Set verbatim=1" est incluse dans l'exemple pour garantir que les espaces ne sont pas automatiquement tronqués avant le démonstration de la fonction ltrim. Voir Verbatim (page 201) pour plus d'informations.*

Résultat

Chaîne	StringLength	LtrimStringLength
def	6	5
abc	10	7

Voir aussi :

p *RTrim (page 1456)*

Mid

Mid() renvoie la partie de la chaîne d'entrée commençant à la position du caractère défini par le deuxième argument, « start », et renvoyant le nombre de caractères spécifié par le troisième argument, « count ». Si « count » est omis, c'est le reste de la chaîne d'entrée qui est renvoyé. Le premier caractère indiqué dans la chaîne d'entrée porte le numéro 1.

Syntaxe :

```
Mid(text, start[, count])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
start	Entier définissant la position du premier caractère de la chaîne text à inclure.
count	Définit la longueur de la chaîne de sortie. S'il est omis, tous les caractères de la position définis par **start** sont inclus.

Exemple : Expressions de graphique

Exemple	Résultat
`Mid('abcdef',3 )`	Renvoie 'cdef'.
`Mid('abcdef',3, 2 )`	Renvoie 'cd'.

Exemple : Script de chargement

```
T1: Load *,  mid(Text,Start) as Mid1,            mid(Text,Start,Count) as Mid2;     Load *
inline [ Text, Start, Count 'abcdef', 3, 2 'abcdef', 2, 3 '210714', 3, 2 '210714', 2, 3 ];
```

Résultat

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *Mid* dans le script de chargement.

Texte	Début	Mid1	Total	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Voir aussi :

p *Index (page 1442)*

Ord

Ord() renvoie le numéro de point de code Unicode du premier caractère de la chaîne d'entrée.

Syntaxe :

```
Ord(text)
```

Type de données renvoyé : entier

Exemples et résultats :

Exemple : Expression de graphique

Exemple	Résultat
`Ord('A')`	Renvoie l'entier 65.
`Ord('Ab')`	Renvoie l'entier 65.

Exemple : Script de chargement

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Résultat :

Chinois	Occidental	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() renvoie une chaîne composée des caractères contenus dans la chaîne d'entrée ('text'), à l'exception des caractères inclus dans le deuxième argument ('remove_chars').

Syntaxe :

```
PurgeChar(text, remove_chars)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
remove_chars	Chaîne contenant les caractères figurant dans text à supprimer.

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
`PurgeChar ( 'a1b2c3','123' )`	Renvoie 'abc'.
`PurgeChar ( 'a1b2c3','312' )`	Renvoie 'abc'.

Exemple : Script de chargement

```
T1: Load  *,  purgechar(String1, String2) as PurgeChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Résultats

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *PurgeChar* dans le script de chargement.

String1	String2	PurgeChar
a1b2c3	123	abc

Voir aussi :

p *KeepChar (page 1446)*

Repeat

Repeat() forme une chaîne composée de la chaîne d'entrée répétée autant de fois que le nombre défini par le deuxième argument.

Syntaxe :

```
Repeat(text[, repeat_count])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
repeat_ count	Définit le nombre de fois que les caractères figurant dans la chaîne **text** doivent être séparés dans la chaîne de sortie.

Exemple : Expression de graphique

Exemple	Résultat
`Repeat( ' * ', rating )` when **rating** = 4	Renvoie '****'.

Exemple : Script de chargement

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Résultat

Chaîne	Répéter
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() renvoie une chaîne après avoir remplacé toutes les occurrences d'une sous-chaîne donnée dans la chaîne d'entrée par une autre sous-chaîne. La fonction n'est pas récursive et fonctionne de gauche à droite.

Syntaxe :

```
Replace(text, from_str, to_str)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
from_str	Chaîne qui peut figurer une ou plusieurs fois dans la chaîne d'entrée **text**.
to_str	Chaîne destinée à remplacer toutes les occurrences de **from_str** dans la chaîne **text**.

Exemples et résultats :

Exemple	Résultat
`Replace('abccde','cc','xyz')`	Renvoie 'abxyzde'.

Voir aussi :

Right

Right() renvoie une chaîne composée des derniers caractères (situés à l'extrémité droite) de la chaîne d'entrée et dont le nombre de caractères est déterminé par le deuxième argument.

Syntaxe :

```
Right(text, count)
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine.
count	Définit le nombre de caractères à inclure en partant de l'extrémité droite de la chaîne **text**.

Exemple : Expression de graphique

Exemple	Résultat
`Right('abcdef', 3)`	Renvoie 'def'.

Exemple : Script de chargement

```
T1: Load  *,  right(Text,Start) as Right;        Load * inline [ Text, Start 'abcdef', 3
'2021-07-14', 4 '2021-07-14', 2 ];
```

Résultat

Table Qlik Sense affichant les résultats obtenus suite à l'utilisation de la fonction *Right* dans le script de chargement.

Texte	Début	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() renvoie la chaîne d'entrée exempte de tout espace de fin.

Syntaxe :

```
RTrim(text)
```

Type de données renvoyé : chaîne

Exemple : Expressions de graphique

Exemple	Résultat
`RTrim( ' abc' )`	Renvoie ' abc'.
`RTrim( 'abc ' )`	Renvoie 'abc'.

Exemple : Script de chargement

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

> *L'instruction "Set verbatim=1" est incluse dans l'exemple pour garantir que les espaces ne sont pas automatiquement tronqués avant le démonstration de la fonction rtrim. Voir Verbatim (page 201) pour plus d'informations.*

Résultat

Chaîne	StringLength	RtrimStringLength
def	6	4
abc	10	6

Voir aussi :

p *LTrim (page 1450)*

SubField

SubField() permet d'extraire des composants de sous-chaîne d'un champ de chaîne parent, où les champs d'enregistrement d'origine se composent de plusieurs parties séparées par un délimiteur.

La fonction **Subfield()** peut s'utiliser, par exemple, pour extraire le prénom et le nom de famille d'une liste d'enregistrements constituée de noms complets, les parties de composant d'un nom de chemin ou encore les données de tables séparées par des virgules.

Si vous utilisez la fonction **Subfield()** dans une instruction **LOAD** en ignorant le paramètre field_no facultatif, un enregistrement complet sera généré pour chaque sous-chaîne. Si plusieurs champs sont chargés à l'aide de la fonction **Subfield()**, les produits cartésiens de toutes les combinaisons sont créés.

Syntaxe :

```
SubField(text, delimiter[, field_no ])
```

Type de données renvoyé : chaîne

Arguments :

Arguments

Argument	Description
text	Chaîne d'origine. Il peut s'agir d'un texte codé de manière irréversible, d'une variable, d'une expansion dollar ou d'une autre expression.
delimiter	Caractère inclus dans la chaîne d'entrée **text** qui divise la chaîne en plusieurs composants.
field_no	Le troisième argument, facultatif, est un entier spécifiant laquelle des sous-chaînes de la chaîne parent **text** doit être renvoyée. Utilisez la valeur 1 pour renvoyer la première sous-chaîne, la valeur 2 pour renvoyer la deuxième sous-chaîne, et ainsi de suite. • Si **field_no** est une valeur positive, les sous-chaînes sont extraites de gauche à droite. • Si **field_no** est une valeur négative, les sous-chaînes sont extraites de droite à gauche.

Il est possible d'utiliser SubField() à la place de combinaisons de fonctions complexes telles que Len(), Right(), Left(), Mid() et d'autres fonctions de chaîne.

Exemples : Script et expressions de graphique via SubField

Exemples - script et expressions de graphique

Exemples de base

Exemple	Résultat
`SubField(S, ';' ,2)`	Renvoie 'cde' si **S** correspond à 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Renvoie une chaîne vide si **S** est une chaîne vide.

Exemple	Résultat
`SubField(S, ';' ,1)`	Renvoie une chaîne vide si **S** correspond à ';'.
Imaginons que vous ayez une variable contenant le nom de chemin d'accès vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	Dans un graphique de type texte et image, vous pouvez ajouter une mesure telle que : `SubField(vMyPath, '\',-3)`, ce qui aboutit à 'Qlik', car il s'agit de la troisième sous-chaîne en partant de l'extrémité droite de la variable vMyPath.

Exemple de script 1

Script de chargement

Chargez les expressions de script et les données suivantes dans l'éditeur de chargement de données.

```
FullName:                                    LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames:  Lo
(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table
FullName;
```

Création d'une visualisation

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Name**, **FirstName** et **SurName**.

Résultat

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Explication

La fonction **SubField()** extrait la première sous-chaîne de **Name** en définissant l'argument **field_no** sur 1. Étant donné que la valeur de **field_no** est positive, un ordre de gauche à droite est suivi pour l'extraction de la sous-chaîne. Un deuxième appel de fonction extrait la deuxième sous-chaîne en définissant l'argument **field _no** sur -1, ce qui extrait la sous-chaîne en suivant un ordre de droite à gauche.

Exemple de script 2

Script de chargement

Chargez les expressions de script et les données suivantes dans l'éditeur de chargement de données.

```
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 '|');
```

Création d'une visualisation

Créez une visualisation de table dans une feuille Qlik Sense dotée des dimensions **Instrument**, **Player** et **Project**.

Résultat

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Explication

Cet exemple montre comment l'utilisation de plusieurs instances de la fonction **Subfield()**, chacune omettant le paramètre field_no, à partir de la même instruction **LOAD** crée des produits cartésiens de toutes les combinaisons. L'option **DISTINCT** est utilisée pour éviter de créer des enregistrements en double.

SubStringCount

SubStringCount() renvoie le nombre d'occurrences de la sous-chaîne spécifiée dans le texte de la chaîne d'entrée. Si aucun caractère ne correspond, la fonction renvoie 0.

Syntaxe :

```
SubStringCount(text, sub_string)
```

Type de données renvoyé : entier

Arguments :

Argument	Description
text	Chaîne d'origine.
sub_string	Chaîne qui peut figurer une ou plusieurs fois dans la chaîne d'entrée **text**.

Exemple : Expressions de graphique

Exemple	Résultat
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Renvoie 2.
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Renvoie 0.

Exemple : Script de chargement

```
T1: Load *,  substringcount(upper(Strings),'AB') as SubStringCount_AB; Load * inline [ Strings
ABC:DEF:GHI:AB:CD:EF:GH aB/cd/ef/gh/Abc/abandoned ];
```

Résultat

Chaînes	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() renvoie le texte de la chaîne d'entrée figurant entre les caractères spécifiés comme délimiteurs.

Syntaxe :

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Type de données renvoyé : chaîne

Arguments :

Argument	Description
text	Chaîne d'origine.
delimiter1	Spécifie le premier caractère (ou la première chaîne) délimiteur à rechercher dans la chaîne **text**.
delimiter2	Spécifie le deuxième caractère (ou la deuxième chaîne) délimiteur à rechercher dans la chaîne **text**.
n	Définit l'occurrence de la paire de délimiteurs à l'intérieur de laquelle la recherche doit porter. Par exemple, une valeur de 2 renvoie les caractères compris dans la deuxième occurrence de delimiter1 et la deuxième occurrence de delimiter2.

Exemple : Expressions de graphique

Exemple	Résultat
`TextBetween('<abc>', '<', '>')`	Renvoie 'abc'.

Exemple	Résultat
`TextBetween('<abc><de>', '<', '>',2)`	Renvoie 'de'.
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Les deux exemples renvoient la valeur NULL. Si la chaîne ne contient aucun des délimiteurs, la valeur NULL est renvoyée.
`TextBetween('<>', '<', '>')`	Renvoie une chaîne de longueur zéro.
`TextBetween('<abc>', '<', '>', 2)`	Renvoie NULL, car n est supérieur au nombre d'occurrences des délimiteurs.

Exemple : Script de chargement

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Résultat

Texte	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() renvoie la chaîne d'entrée exempte de tout espace de début et de fin.

Syntaxe :

```
Trim(text)
```

Type de données renvoyé : chaîne

Exemples et résultats :

Exemple : Expression de graphique

Exemple	Résultat
`Trim( ' abc' )`	Renvoie 'abc'.
`Trim( 'abc ' )`	Renvoie 'abc'.
`Trim( ' abc ' )`	Renvoie 'abc'.

Exemple : Script de chargement

```
Set verbatim=1;                                          T1: Load *, len(TrimString) as TrimStringLength;
(String) as TrimString; Load *, len(String) as StringLength;              Load * inline [
```

```
String '   abc    ' ' def  '](delimiter is '\t');
```

L'instruction "Set verbatim=1" est incluse dans l'exemple pour garantir que les espaces ne sont pas automatiquement tronqués avant le démonstration de la fonction trim. Voir Verbatim (page 201) pour plus d'informations.

Résultat :

Chaîne	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() convertit les caractères de la chaîne d'entrée en lettres majuscules pour tous les caractères de texte de l'expression. Les nombres et les symboles sont ignorés.

Syntaxe :

```
Upper(text)
```

Type de données renvoyé : chaîne

Exemple : Expression de graphique

Exemple	Résultat
`Upper(' abcD')`	Renvoie 'ABCD'.

Exemple : Script de chargement

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Résultat

Chaîne	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Fonctions système

Les fonctions système comprennent des fonctions permettant d'accéder aux propriétés du système, des appareils et périphériques, et des applications Qlik Sense.

Vue d'ensemble des fonctions système

Certaines fonctions sont décrites plus en détail après la vue d'ensemble. Dans ce cas, il vous suffit de cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Author()

Cette fonction renvoie une chaîne contenant les propriétés de l'auteur de l'application active. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

> *Il n'est pas possible de définir les propriétés de l'auteur dans la version actuelle de Qlik Sense. Si vous migrez un document QlikView, les propriétés de l'auteur seront conservées.*

ClientPlatform()

Cette fonction renvoie la chaîne de l'agent utilisateur du navigateur client. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

Exemple :

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Cette fonction renvoie une chaîne contenant le nom de l'ordinateur tel que renvoyé par le système d'exploitation. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

> *Si le nom de l'ordinateur comprend plus de 15 caractères, la chaîne ne contiendra que les 15 premiers.*

```
ComputerName( )
```

DocumentName

Cette fonction renvoie une chaîne contenant le nom de l'application Qlik Sense active, sans le chemin d'accès mais avec l'extension. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
DocumentName( )
```

DocumentPath

Cette fonction renvoie une chaîne contenant le chemin d'accès complet à l'application Qlik Sense active. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
DocumentPath( )
```

Cette fonction n'est pas prise en charge en mode standard.

DocumentTitle

Cette fonction renvoie une chaîne contenant le titre de l'application Qlik Sense active. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
DocumentTitle( )
```

EngineVersion

Cette fonction renvoie la version complète du moteur Qlik Sense sous la forme d'une chaîne.

```
EngineVersion ()
```

GetCollationLocale

Cette fonction de script renvoie le nom de culture des paramètres régionaux de classement utilisés. Si la variable CollationLocale n'a pas été définie, ce sont les paramètres régionaux actifs de l'ordinateur de l'utilisateur qui sont renvoyés.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() renvoie le nom de la dimension. **Index** désigne un entier facultatif qui indique la dimension à renvoyer.

```
GetObjectField - fonction de graphique([index])
```

GetRegistryString

Cette fonction renvoie la valeur d'une clé dans le registre de Windows. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
GetRegistryString(path, key)
```

Cette fonction n'est pas prise en charge en mode standard.

IsPartialReload

Cette fonction renvoie -1 (True) si le rechargement en cours est partiel ; sinon, elle renvoie 0 (False).

```
IsPartialReload ()
```

OSUser

Cette fonction renvoie une chaîne contenant le nom de l'utilisateur actuellement connecté. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
OSUser( )
```

Dans Qlik Sense Desktop et Qlik Sense Mobile Client Managed, cette fonction renvoie toujours « Personal\Me ».

ProductVersion

Cette fonction renvoie la version complète de Qlik Sense et le numéro de compilation sous la forme d'une chaîne.

Cette fonction est dépréciée et remplacée par **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime

Cette fonction renvoie un horodatage correspondant à la fin du dernier chargement de données. Peut s'utiliser aussi bien dans le script de chargement de données que dans une expression de graphique.

```
ReloadTime( )
```

StateName

StateName() renvoie le nom de l'état alternatif de la visualisation dans laquelle il est utilisé. StateName peut servir, par exemple, à créer des visualisations comportant du texte et des couleurs dynamiques afin de refléter le changement d'état d'une visualisation. Cette fonction peut s'utiliser dans les expressions de graphique, mais elle ne permet pas de déterminer l'état auquel l'expression fait référence.

```
StateName - fonction de graphique()
```

EngineVersion

Cette fonction renvoie la version complète du moteur Qlik Sense sous la forme d'une chaîne.

Syntaxe :

```
EngineVersion()
```

IsPartialReload

Cette fonction renvoie -1 (True) si le rechargement en cours est partiel ; sinon, elle renvoie 0 (False).

Syntaxe :

```
IsPartialReload()
```

ProductVersion

Cette fonction renvoie la version complète de Qlik Sense et le numéro de compilation sous la forme d'une chaîne. Cette fonction est dépréciée et remplacée par **EngineVersion()**.

Syntaxe :

```
ProductVersion()
```

StateName - fonction de graphique

StateName() renvoie le nom de l'état alternatif de la visualisation dans laquelle il est utilisé. StateName peut servir, par exemple, à créer des visualisations comportant du texte et des

couleurs dynamiques afin de refléter le changement d'état d'une visualisation. Cette fonction peut s'utiliser dans les expressions de graphique, mais elle ne permet pas de déterminer l'état auquel l'expression fait référence.

Syntaxe :

```
StateName ()
```

Example 1:

Texte dynamique

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Couleurs dynamiques

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Fonctions de table

Les fonctions de table renvoient des informations sur la table de données en cours de lecture. Si aucun nom de table n'est spécifié et que la fonction est utilisée dans une instruction **LOAD**, c'est la table active qui est prise en compte.

Toutes les fonctions s'utilisent dans le script de chargement de données tandis que seule **NoOfRows** est admise dans les expressions de graphique.

Vue d'ensemble des fonctions de table

Certaines fonctions sont décrites plus en détail après la vue d'ensemble. Dans ce cas, il vous suffit de cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

FieldName

La fonction de script **FieldName** renvoie le nom du champ portant le numéro indiqué dans une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

```
FieldName (field_number ,table_name)
```

FieldNumber

La fonction de script **FieldNumber** renvoie le numéro d'un champ donné dans une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

La fonction de script **NoOfFields** renvoie le nombre de champs d'une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

```
NoOfFields (table_name)
```

NoOfRows

La fonction **NoOfRows** renvoie le nombre de lignes (d'enregistrements) d'une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

```
NoOfRows (table_name)
```

NoOfTables

Cette fonction de script renvoie le nombre de tables précédemment chargées.

```
NoOfTables()
```

TableName

Cette fonction de script renvoie le nom de la table portant le numéro indiqué.

```
TableName(table_number)
```

TableNumber

Cette fonction de script renvoie le numéro de la table spécifiée. La première table porte le numéro 0.

Si table_name n'existe pas, NULL est renvoyé.

```
TableNumber(table_name)
```

Exemple :

Dans cet exemple, nous souhaitons créer une table contenant des informations sur les tables et les champs qui ont été chargés.

Nous commençons par charger des échantillons de données. Cela a pour effet de créer les deux tables qui serviront à illustrer les fonctions de table décrites dans cette section.

```
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 ;
```

Ensuite, nous passons à l'itération au sein des tables déjà chargées, à l'aide de la fonction **NoOfTables**, puis au sein des champs de chaque table, avec la fonction **NoOfFields**. Nous chargeons ensuite les informations au moyen des fonctions de table.

```
//Iterate through the loaded tables
For t = 0 to NoOfTables() - 1

//Iterate through the fields of table
 For f = 1 to NoOfFields(TableName($(t)))
  Tables:
  Load
   TableName($(t)) as Table,
   TableNumber(TableName($(t))) as TableNo,
   NoOfRows(TableName($(t))) as TableRows,
   FieldName($(f),TableName($(t))) as Field,
   FieldNumber(FieldName($(f),TableName($(t))),TableName($(t))) as FieldNo
   Autogenerate 1;
 Next f
Next t;
```

La table résultante Tables a l'aspect suivant :

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

La fonction de script **FieldName** renvoie le nom du champ portant le numéro indiqué dans une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

Syntaxe :

```
FieldName(field_number ,table_name)
```

Arguments :

Arguments

Argument	Description
field_number	Numéro de champ du champ à référencer.
table_name	Table contenant le champ à référencer.

Exemple :

```
LET a = FieldName(4,'tab1');
```

FieldNumber

La fonction de script **FieldNumber** renvoie le numéro d'un champ donné dans une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

Syntaxe :

```
FieldNumber(field_name ,table_name)
```

Arguments :

Arguments

Argument	Description
field_name	Nom du champ.
table_name	Nom de la table contenant le champ.

Si le champ field_name ne figure pas dans l'argument table_name ou que l'argument table_name n'existe pas, la fonction renvoie 0.

Exemple :

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

La fonction de script **NoOfFields** renvoie le nombre de champs d'une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

Syntaxe :

```
NoOfFields(table_name)
```

Arguments :

Arguments

Argument	Description
table_name	Nom de la table.

Exemple :

```
LET a = NoOfFields('tab1');
```

NoOfRows

La fonction **NoOfRows** renvoie le nombre de lignes (d'enregistrements) d'une table déjà chargée. Si la fonction est utilisée dans une instruction **LOAD**, elle ne doit pas faire référence à la table en cours de chargement.

Syntaxe :

```
NoOfRows(table_name)
```

Arguments :

Arguments

Argument	Description
table_name	Nom de la table.

Exemple :

```
LET a = NoOfRows('tab1');
```

5.27 Fonctions trigonométriques et hyperboliques

Cette section décrit les fonctions permettant de réaliser des opérations trigonométriques et hyperboliques. Dans toutes les fonctions, les arguments sont des expressions fournissant des angles mesurés en radians, où **x** doit être interprété comme un nombre réel.

Tous les angles sont mesurés en radians.

Elles s'utilisent toutes aussi bien dans le script de chargement de données que dans les expressions de graphique.

cos
Cosinus de **x**. Le résultat est un nombre compris entre -1 et 1.

```
cos( x )
```

acos
Cosinus inverse de **x**. La fonction est uniquement définie si -1≤**x**≤1. Le résultat est un nombre compris entre 0 et π.

```
acos( x )
```

sin
Sinus de **x**. Le résultat est un nombre compris entre -1 et 1.

```
sin( x )
```

asin

Sinus inverse de **x**. La fonction est uniquement définie si -1≤**x**≤1. Le résultat est un nombre compris entre -π/2 et π/2.

```
asin( x )
```

tan

Tangente de **x**. Le résultat est un nombre réel.

```
tan( x )
```

atan

Tangente inverse de **x**. Le résultat est un nombre compris entre - π/2 et π/2.

```
atan( x )
```

atan2

Généralisation bidimensionnelle de la fonction de tangente inverse. Renvoie l'angle formé entre l'origine et le point représenté par les coordonnées **x** et **y**. Le résultat est un nombre compris entre - π et + π.

```
atan2( y,x )
```

cosh

Cosinus hyperbolique de **x**. Le résultat est un nombre réel positif.

```
cosh( x )
```

sinh

Sinus hyperbolique de **x**. Le résultat est un nombre réel.

```
sinh( x )
```

tanh

Tangente hyperbolique de **x**. Le résultat est un nombre réel.

```
tanh( x )
```

acosh

Cosinus hyperbolique inverse de **x**. Le résultat est un nombre réel positif.

```
acosh( x )
```

asinh

Sinus hyperbolique inverse de **x**. Le résultat est un nombre réel.

```
asinh( x )
```

atanh

Tangente hyperbolique inverse de **x**. Le résultat est un nombre réel.

```
atanh( x )
```

Exemples :

Le code de script suivant charge un échantillon de table, puis une table contenant les opérations trigonométriques et hyperboliques calculées sur les valeurs.

```
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 Restrictions d'accès au système de fichiers

Pour des raisons de sécurité, lorsque Qlik Sense est en mode standard, il ne prend pas en charge les chemins d'accès dans le script de chargement de données, ni les fonctions et variables qui exposent le système de fichiers.

Cependant, comme les chemins d'accès au système de fichiers étaient pris en charge par QlikView, il est possible de désactiver le mode standard au profit du mode hérité afin de pouvoir réutiliser les scripts de chargement QlikView.

La désactivation du mode standard représente un risque de sécurité potentiel en exposant le système de fichiers.

Désactivation du mode standard (page 1480)

6.1 Aspects liés à la sécurité lors d'une connexion à des connexions de données ODBC et OLE DB basées sur des fichiers

Les connexions de données ODBC et OLE DB utilisant des pilotes basés sur des fichiers présentent le chemin d'accès au fichier de données connecté dans la chaîne de connexion. Le chemin d'accès peut être présenté lorsque la connexion est éditée, dans la boîte de dialogue de sélection de données, ou dans certaines requêtes SQL. C'est le cas en mode standard comme en mode hérité.

Si la présentation du chemin d'accès au fichier de données pose problème, il est recommandé de se connecter au fichier de données au moyen d'une connexion de données de type dossier, si cela est possible.

6.2 Limitations inhérentes au mode standard

L'utilisation de certaines instructions, variables et fonctions est impossible ou limitée en mode standard. L'emploi d'instructions non prises en charge dans le script de chargement de données génère une erreur au moment de l'exécution du script. Les messages d'erreur sont consignés dans le fichier journal du script. En revanche, l'utilisation de variables et fonctions non prises en charge ne génère pas de messages d'erreur ni d'entrées dans le fichier journal. Elle entraîne le renvoi de la valeur NULL.

Pendant l'édition du script de chargement de données, rien ne vous indique qu'une variable, instruction ou fonction donnée n'est pas prise en charge.

Variables système

Variables système

Variable	Mode standard	Mode hérité	Définition
Floppy	Non pris en charge	Pris en charge	Renvoie la lettre du premier lecteur de disquette trouvé, normalement *a:*.
CD	Non pris en charge	Pris en charge	Renvoie la lettre du premier lecteur de CD-ROM trouvé. Si aucun lecteur de CD-ROM n'est trouvé, la valeur *c:* est renvoyée.
QvPath	Non pris en charge	Pris en charge	Renvoie la chaîne de navigation jusqu'à l'exécutable Qlik Sense.
QvRoot	Non pris en charge	Pris en charge	Renvoie le répertoire racine de l'exécutable Qlik Sense.
QvWorkPath	Non pris en charge	Pris en charge	Renvoie la chaîne de navigation jusqu'à l'application Qlik Sense active.
QvWorkRoot	Non pris en charge	Pris en charge	Renvoie le répertoire racine de l'application Qlik Sense active.
WinPath	Non pris en charge	Pris en charge	Renvoie la chaîne de navigation jusqu'à Windows.
WinRoot	Non pris en charge	Pris en charge	Renvoie le répertoire racine de Windows.

Variable	Mode standard	Mode hérité	Définition
$(include=...)	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	La variable **Include/Must_Include** spécifie un fichier qui contient le texte à inclure dans le script et à évaluer comme code de script. Elle n'est pas utilisée pour ajouter des données. Il est possible de stocker des parties du code de script dans un fichier texte distinct afin de les réutiliser dans d'autres applications. Il s'agit d'une variable définie par l'utilisateur.

Instructions de script normales

Instructions de script normales

Instruction	Mode standard	Mode hérité	Définition
Binary	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **binary** s'utilise pour charger des données issues d'une autre application.
Connect	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **CONNECT** permet de définir l'accès de Qlik Sense à une base de données générale via l'interface OLE DB/ODBC. Pour ODBC, la source de données doit d'abord être spécifiée à l'aide de l'administrateur ODBC.

Instruction	Mode standard	Mode hérité	Définition
Directory	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **Directory** définit le répertoire dans lequel le programme doit rechercher les fichiers de données dans les instructions **LOAD** ultérieures, jusqu'à ce qu'une nouvelle instruction **Directory** soit définie.
Execute	Non pris en charge	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **Execute** permet d'exécuter d'autres programmes pendant que Qlik Sense est en train de charger des données. Elle s'utilise, par exemple, pour effectuer des conversions nécessaires.
LOAD from ...	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **LOAD** charge des champs à partir d'un fichier, de données définies dans le script, d'une table déjà chargée, d'une page Web, du résultat d'une instruction **SELECT** ultérieure ou via la génération automatique de données.
Store into ...	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	L'instruction **Store** crée un fichier QVD, CSV ou text.

Instructions de contrôle de script

Instructions de contrôle de script

Instruction	Mode standard	Mode hérité	Définition
For each... filelist mask/dirlist mask	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque Résultat renvoyé : connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers Résultat renvoyé : connexion de type bibliothèque ou chemin d'accès au système de fichiers, selon les données d'entrée	La syntaxe filelist mask génère une liste de tous les fichiers, séparés par des virgules, qui se trouvent dans le répertoire actif et qui correspondent à l'instruction **filelist mask**. La syntaxe **dirlist mask** génère une liste de tous les répertoires, séparés par des virgules, qui se trouvent dans le répertoire actif et qui correspondent au masque de nom de répertoire.

Fonctions de fichier

Fonctions de fichier

Fonction	Mode standard	Mode hérité	Définition
Attribute()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Renvoie sous forme de texte la valeur des balises méta de différents fichiers multimédia.
ConnectString()	Résultat renvoyé : nom de la connexion à la bibliothèque	Nom de la connexion à la bibliothèque ou connexion réelle, selon les données entrées	Renvoie la chaîne connect active pour les connexions ODBC ou OLE DB.
FileDir()	Résultat renvoyé : connexion à la bibliothèque	Résultat renvoyé : connexion de type bibliothèque ou chemin d'accès au système de fichiers, selon les données d'entrée	La fonction **FileDir** renvoie une chaîne contenant le chemin d'accès au répertoire dans lequel figure le fichier de table en cours de lecture.

Fonction	Mode standard	Mode hérité	Définition
FilePath()	Résultat renvoyé : connexion à la bibliothèque	Résultat renvoyé : connexion de type bibliothèque ou chemin d'accès au système de fichiers, selon les données d'entrée	La fonction **FilePath** renvoie une chaîne contenant le chemin d'accès complet au fichier de table en cours de lecture.
FileSize()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	La fonction **FileSize** renvoie un entier contenant la taille en octets du fichier filename ou, si aucun argument filename n'est spécifié, du fichier de table en cours de lecture.
FileTime()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	La fonction **FileTime** renvoie un horodatage au format UTC pour la date et l'heure de la dernière modification apportée au fichier filename. Si aucun argument filename n'est spécifié, la fonction se réfère au fichier de table en cours de lecture.
GetFolderPath()	Non pris en charge	Résultat renvoyé : chemin absolu	La fonction **GetFolderPath** renvoie la valeur de la fonction Microsoft Windows *SHGetFolderPath*. Cette fonction utilise comme données d'entrée le nom d'un dossier Microsoft Windows et renvoie le chemin d'accès complet au dossier.

Fonction	Mode standard	Mode hérité	Définition
QvdCreateTime()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Cette fonction de script renvoie l'horodatage de l'en-tête XML à partir d'un fichier QVD, le cas échéant. Dans le cas contraire, la valeur NULL est renvoyée. Dans l'horodatage, l'heure est indiquée dans le fuseau horaire UTC.
QvdFieldName()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Cette fonction de script renvoie le nom du numéro de champ **fieldno** contenu dans un fichierQVD. S'il n'existe pas de champ, NULL est renvoyé.
QvdNoOfFields()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Cette fonction de script renvoie le nombre de champs contenus dans un fichier QVD.
QvdNoOfRecords()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Cette fonction de script renvoie le nombre d'enregistrements contenus dans un fichier QVD.
QvdTableName()	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque	Entrée prise en charge : Chemin d'accès via la connexion à la bibliothèque ou le système de fichiers	Cette fonction de script renvoie le nom de la table stockée dans un fichier QVD.

Fonctions système

Fonctions système

Fonction	Mode standard	Mode hérité	Définition
DocumentPath()	Non pris en charge	Résultat renvoyé : chemin absolu	Cette fonction renvoie une chaîne contenant le chemin d'accès complet à l'application Qlik Sense active.
GetRegistryString()	Non pris en charge	Pris en charge	Renvoie la valeur d'une clé de registre existante avec un chemin de registre donné. Cette fonction peut s'utiliser aussi bien dans les graphiques que dans les scripts.

6.3 Désactivation du mode standard

Vous pouvez désactiver le mode standard, c'est-à-dire définir le mode hérité, afin de réutiliser des scripts de chargement QlikView qui font référence à des chemins de fichier absolus ou relatifs de même qu'à des connexions de bibliothèque.

La désactivation du mode standard représente un risque de sécurité potentiel en exposant le système de fichiers.

Qlik Sense

Dans Qlik Sense, le mode standard peut être désactivé via la propriété **Standard mode** (Mode standard) de la console QMC.

Qlik Sense Desktop

Dans Qlik Sense Desktop, vous pouvez définir le mode standard/hérité dans le fichier *Settings.ini*.

Si vous avez installé Qlik Sense Desktop à l'emplacement d'installation par défaut, alors le fichier *Settings.ini* se trouve sous *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Si vous avez installé Qlik Sense Desktop dans un dossier que vous avez sélectionné, alors le fichier *Settings.ini* se trouve dans le dossier *Engine* du chemin d'installation.

Procédez comme suit :

1. Ouvrez le fichier *Settings.ini* dans un éditeur de texte.
2. Remplacez la ligne *StandardReload=1* par *StandardReload=0*.
3. Enregistrez le fichier et lancez Qlik Sense Desktop.

Qlik Sense Desktop fonctionne à présent en mode hérité.

Paramètres

Les paramètres disponibles pour l'option StandardReload sont les suivants :

- 1 (mode standard)
- 0 (mode hérité)

6 Scriptage graphique

Lors de la modification des données du graphique, vous utilisez un sous-ensemble du script Qlik Sense, constitué d'une certain nombre d'instructions. Une instruction peut désigner soit une instruction de script normale, soit une instruction de contrôle de script. Certaines instructions peuvent être précédées de préfixes.

Les instructions normales servent généralement à manipuler des données d'une manière ou d'une autre. Ces instructions peuvent être écrites sur autant de lignes de script que nécessaire et doivent toujours se terminer par un point-virgule « ; ».

Les instructions de contrôle sont généralement utilisées pour contrôler le flux de l'exécution du script. Chaque clause d'une instruction de contrôle doit tenir sur une ligne de script et peut se terminer par un point-virgule ou une fin de ligne.

Il est possible d'appliquer des préfixes aux instructions normales pertinentes mais jamais aux instructions de contrôle.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

Dans cette section, vous trouverez une liste alphabétique de l'ensemble des instructions de script, instructions de contrôle et préfixes disponibles dans le sous-ensemble du script utilisé lors de la modification des données d'un graphique.

6.4 Instructions de contrôle

Lors de la modification des données du graphique, vous utilisez un sous-ensemble du script Qlik Sense, constitué d'une certain nombre d'instructions. Une instruction peut désigner soit une instruction de script normale, soit une instruction de contrôle de script.

Les instructions de contrôle sont généralement utilisées pour contrôler le flux de l'exécution du script. Chaque clause d'une instruction de contrôle doit tenir sur une ligne de script et peut se terminer par un point-virgule ou une fin de ligne.

Aucun préfixe n'est jamais appliqué aux instructions de contrôle.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules.

Vue d'ensemble des instructions de contrôle d'un modificateur de graphique

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Call

L'instruction de contrôle **call** appelle une sous-routine qui doit être définie par une instruction **sub** précédente.

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

Do..loop

L'instruction de contrôle **do..loop** est une construction d'itération de script qui exécute une ou plusieurs instructions jusqu'à ce qu'une condition logique soit remplie.

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

End

Le mot-clé de script **End** permet de fermer les clauses **If**, **Sub** et **Switch**.

Exit

Le mot-clé de script **Exit** fait partie de l'instruction **Exit Script**, mais il peut aussi être employé pour quitter les clauses **Do**, **For** ou **Sub**.

Exit script

Cette instruction de contrôle arrête l'exécution du script. Elle peut être insérée n'importe où dans le script.

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

For..next

L'instruction de contrôle **for..next** est une construction d'itération de script avec compteur. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la variable du compteur entre les limites inférieure et supérieure spécifiées.

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

For each ..next

L'instruction de contrôle **for each..next** est une construction d'itération de script qui exécute une ou plusieurs instructions pour chaque valeur d'une liste de valeurs séparées par des virgules. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la liste.

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

If..then

L'instruction de contrôle **if..then** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon une ou plusieurs conditions logiques.

> *Comme l'instruction **if..then** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**if..then**, **elseif..then**, **else** et **end if**) ne peut s'étendre sur plusieurs lignes.*

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

Next

Le mot-clé de script **Next** permet de fermer les boucles **For**.

Sub

L'instruction de contrôle **sub..end sub** définit une sous-routine qui peut être appelée à partir d'une instruction **call**.

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

Switch

L'instruction de contrôle **switch** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon la valeur d'une expression.

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

To

Le mot-clé de script **To** s'utilise dans plusieurs instructions de script.

Call

L'instruction de contrôle **call** appelle une sous-routine qui doit être définie par une instruction **sub** précédente.

Syntaxe :

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

Arguments :

Arguments

Argument	Description
name	Nom de la sous-routine.
paramlist	Liste des paramètres à envoyer à la sous-routine, séparés par des virgules. Chaque élément de la liste peut correspondre à un nom de champ, une variable ou une expression arbitraire.

La sous-routine appelée par une instruction **call** doit être définie auparavant dans l'exécution du script par une instruction **sub**.

Les paramètres sont copiés dans la sous-routine et, si le paramètre de l'instruction **call** désigne une variable au lieu d'une expression, il est recopié et supprimé à la fermeture de la sous-routine.

Limitations :

- Comme l'instruction **call** est une instruction de contrôle et qu'elle se termine donc soit par un point-virgule, soit par un caractère de fin de ligne, elle ne doit pas s'étendre sur plusieurs lignes.
- Lorsque vous définissez une sous-routine avec `Sub..end sub` à l'intérieur d'une instruction de contrôle, par exemple `if..then`, vous pouvez uniquement appeler la sous-routine depuis la même instruction de contrôle.

Do..loop

L'instruction de contrôle **do..loop** est une construction d'itération de script qui exécute une ou plusieurs instructions jusqu'à ce qu'une condition logique soit remplie.

Syntaxe :

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

*Comme l'instruction **do..loop** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**do**,**exit do** et **loop**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.
while / until	Les clauses conditionnelles **while** ou **until** ne doivent figurer qu'une fois dans une instruction **do..loop**, soit après **do**, soit après **loop**. Chaque condition n'est interprétée que la première fois, mais elle est évaluée à chaque fois que le script la rencontre dans la boucle.
exit do	Si une clause **exit do** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **loop** indiquant la fin de la boucle. Il est possible de rendre une clause **exit do** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

End

Le mot-clé de script **End** permet de fermer les clauses **If**, **Sub** et **Switch**.

Exit

Le mot-clé de script **Exit** fait partie de l'instruction **Exit Script**, mais il peut aussi être employé pour quitter les clauses **Do**, **For** ou **Sub**.

Exit script

Cette instruction de contrôle arrête l'exécution du script. Elle peut être insérée n'importe où dans le script.

Syntaxe :

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

Comme l'instruction **exit script** est une instruction de contrôle et qu'elle se termine donc soit par un point-virgule, soit par un caractère de fin de ligne, elle ne doit pas s'étendre sur plusieurs lignes.

Arguments :

Arguments

Argument	Description
condition	Expression logique dont l'évaluation a pour résultat True ou False.
when / unless	Il est possible de rendre une instruction **exit script** conditionnelle par l'utilisation facultative d'une clause **when** ou **unless**.

Exemples :

```
//Exit script
Exit Script;

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

For..next

L'instruction de contrôle **for..next** est une construction d'itération de script avec compteur. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la variable du compteur entre les limites inférieure et supérieure spécifiées.

Syntaxe :

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

Les expressions *expr1*, *expr2* et *expr3* ne sont évaluées que la première fois que le script entre dans la boucle. Il est possible de modifier la valeur de la variable counter à l'aide d'instructions placées à l'intérieur de la boucle, mais ce n'est pas une bonne méthode de programmation.

Si une clause **exit for** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **next** indiquant la fin de la boucle. Il est possible de rendre une clause **exit for** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

> *Comme l'instruction **for..next** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**for..to..step**,**exit for** et **next**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
counter	Nom de variable. Si l'argument *counter* est spécifié après **next**, il doit s'agir du même nom de variable que celui qui se trouve après le **for** correspondant.
expr1	Expression qui détermine la première valeur de la variable *counter* pour laquelle la boucle doit être exécutée.
expr2	Expression qui détermine la dernière valeur de la variable *counter* pour laquelle la boucle doit être exécutée.

Argument	Description
expr3	Expression qui détermine la valeur de l'incrément de la variable *counter* lors de chaque exécution de la boucle.
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

For each..next

L'instruction de contrôle **for each..next** est une construction d'itération de script qui exécute une ou plusieurs instructions pour chaque valeur d'une liste de valeurs séparées par des virgules. Les instructions comprises entre **for** et **next** à l'intérieur de la boucle sont exécutées pour chaque valeur de la liste.

Syntaxe :
Une syntaxe spéciale permet de générer des listes comprenant les noms des fichiers et des répertoires contenus dans le répertoire actif.

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

Arguments :

Arguments

Argument	Description
var	Nom de variable de script qui prendra une nouvelle valeur à partir de la liste lors de chaque exécution de la boucle. Si l'argument **var** est spécifié après **next**, il doit s'agir du même nom de variable que celui qui se trouve après le **for each** correspondant.

Il est possible de modifier la valeur de la variable **var** à l'aide d'instructions placées à l'intérieur de la boucle, mais ce n'est pas une bonne méthode de programmation.

Si une clause **exit for** se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause **next** indiquant la fin de la boucle. Il est possible de rendre une clause **exit for** conditionnelle par l'utilisation facultative d'un suffixe **when** ou **unless**.

*Comme l'instruction **for each..next** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses trois clauses possibles (**for each**,**exit for** et **next**) ne peut s'étendre sur plusieurs lignes.*

Syntaxe :

```
list := item { , item }
```

```
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Arguments

Argument	Description
constant	Tout nombre ou toute chaîne. Veuillez noter qu'une chaîne écrite directement dans le script doit être placée entre guillemets simples. Une chaîne non mise entre guillemets simples est interprétée comme une variable ; la valeur de la variable lui est ensuite appliquée. Il est inutile de placer les nombres entre guillemets simples.
expression	Expression arbitraire.
mask	Masque de nom de fichier ou de dossier pouvant inclure n'importe quel caractère de nom de fichier valide, ainsi que les caractères génériques standard, * et ?. Vous pouvez utiliser des chemins d'accès absolus ou des chemins d'accès lib://.
condition	Expression logique dont l'évaluation a pour résultat True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.
filelist mask	Cette syntaxe génère une liste de tous les fichiers, séparés par des virgules, qui se trouvent dans le répertoire actif et qui correspondent au masque de nom de fichier. *Cet argument prend uniquement en charge les connexions aux bibliothèques en mode standard.*
dirlist mask	Cette syntaxe génère une liste de tous les dossiers, séparés par des virgules, qui se trouvent dans le dossier actif et qui correspondent au masque de nom de dossier. *Cet argument prend uniquement en charge les connexions aux bibliothèques en mode standard.*
fieldvaluelist mask	Cette syntaxe itère au sein des valeurs d'un champ déjà chargé dans Qlik Sense.

Le Qlik Connecteurs de fournisseurs de stockage Web et les autres connexions DataFiles ne prennent pas en charge les masques de filtre utilisant les caractères génériques (et ?).*

Example 1: Chargement d'une liste de fichiers

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

Example 2: Création d'une liste de fichiers sur un disque

Dans cet exemple, la liste de tous les fichiers relatifs à Qlik Sense sont chargés dans un dossier.

```
sub DoDir (Root)
   for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'

      for each File in filelist (Root&'/*.' &Ext)

         LOAD
            '$(File)' as Name,
            FileSize( '$(File)' ) as Size,
            FileTime( '$(File)' ) as FileTime
         autogenerate 1;

      next File

   next Ext
   for each Dir in dirlist (Root&'/*' )

      call DoDir (Dir)

   next Dir

end sub

call DoDir ('lib://DataFiles')
```

Example 3: Itération au sein des valeurs d'un champ

Cet exemple itère au sein de la liste des valeurs chargées de champ FIELD et génère un nouveau champ, NEWFIELD. Pour chaque valeur de FIELD, deux enregistrements NEWFIELD sont créés.

```
load * inline [
FIELD
one
two
three
];

FOR Each a in FieldValueList('FIELD')
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

La table résultante a l'aspect suivant :

Example table

NEWFIELD
one-1
one-2

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

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

L'instruction de contrôle **if..then** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon une ou plusieurs conditions logiques.

Les instructions de contrôle sont généralement utilisées pour contrôler le flux d'exécution du script. Dans une expression de graphique, utilisez plutôt la fonction conditionnelle **if**.

Syntaxe :

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

Comme l'instruction **if..then** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**if..then**, **elseif..then**, **else** et **end if**) ne peut s'étendre sur plusieurs lignes.

Arguments :

Arguments

Argument	Description
condition	Expression logique qui peut être évaluée comme True ou False.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Example 1:

```
if a=1 then

      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;

end if
```

Example 2:

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

Example 3:

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

Next

Le mot-clé de script **Next** permet de fermer les boucles **For**.

Sub..end sub

L'instruction de contrôle **sub..end sub** définit une sous-routine qui peut être appelée à partir d'une instruction **call**.

Syntaxe :

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

Les arguments sont copiés dans la sous-routine et, si les paramètres réels correspondants de l'instruction **call** constituent un nom de variable, ils sont recopiés et supprimés à la fermeture de la sous-routine.

Si une sous-routine comporte plus de paramètres formels que ceux réellement transmis par une instruction **call**, les paramètres supplémentaires sont initialisés sur la valeur NULL et peuvent être utilisés comme variables locales dans la sous-routine.

Arguments :

Arguments

Argument	Description
name	Nom de la sous-routine.
paramlist	Liste de noms de variables séparés par des virgules et définissant les paramètres formels de la sous-routine. Ceux-ci peuvent être utilisés comme n'importe quelle variable au sein de la sous-routine.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Limitations :

- Comme l'instruction **sub** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses deux clauses possibles (**sub** et **end sub**) ne peut s'étendre sur plusieurs lignes.
- Lorsque vous définissez une sous-routine avec `Sub..end sub` à l'intérieur d'une instruction de contrôle, par exemple `if..then`, vous pouvez uniquement appeler la sous-routine depuis la même instruction de contrôle.

Example 1:

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

Example 2: - transfert de paramètres

```
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)
```

Le résultat de l'exemple ci-dessus est que, localement, au sein de la sous-routine, A sera initialisé sur 1, B sur 4 et C sur NULL.

Au moment de quitter la sous-routine, la variable globale A obtiendra la valeur 2 (recopiée à partir de la sous-routine). Le deuxième paramètre réel (X+1)*2 ne sera pas recopié, car il ne s'agit pas d'une variable. Enfin, la variable globale C ne sera pas affectée par l'appel de sous-routine.

Switch..case..default..end switch

L'instruction de contrôle **switch** est une construction de sélection de script qui oblige l'exécution du script à s'orienter dans un sens ou dans un autre selon la valeur d'une expression.

Syntaxe :

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

> *Comme l'instruction **switch** est une instruction de contrôle et, en tant que telle, se termine par un point-virgule ou une fin de ligne, aucune de ses quatre clauses possibles (**switch**, **case**, **default** et **end switch**) ne peut s'étendre sur plusieurs lignes.*

Arguments :

Arguments

Argument	Description
expression	Expression arbitraire.
valuelist	Liste de valeurs séparées par des virgules à laquelle la valeur de l'expression sera comparée. L'exécution du script se poursuit avec les instructions du premier groupe rencontré qui comporte dans l'argument valuelist une valeur égale à la valeur de l'expression. Chaque valeur de l'argument valuelist peut désigner une expression arbitraire. Si aucune valeur correspondante n'est trouvée dans une clause **case**, les instructions figurant dans la clause **default** (si celle-ci est spécifiée) sont exécutées.
statements	Tout groupe d'une ou plusieurs instructions de script Qlik Sense.

Exemple :

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

To

Le mot-clé de script **To** s'utilise dans plusieurs instructions de script.

6.5 Préfixes

Il est possible d'appliquer des préfixes aux instructions normales pertinentes mais jamais aux instructions de contrôle.

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

Vue d'ensemble des préfixes d'un modificateur de graphique

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

Add

Il est possible d'ajouter le préfixe **Add** à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier qu'il faut ajouter des enregistrements à une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Add** peut également être utilisé dans

une instruction **Map**.

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

Replace

Le préfixe **Replace** peut être ajouté à n'importe quelle instruction **LOAD** ou **SELECT** du script pour spécifier que la table chargée doit remplacer une autre table. Cela spécifie également que cette instruction doit être exécutée lors d'un chargement partiel. Le préfixe **Replace** peut également être utilisé dans une instruction **Map**.

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

Add

Dans un contexte de modification de graphique, le préfixe **Add** est utilisé avec **LOAD** pour ajouter des valeurs à la table *HC1*, représentant l'hypercube calculé par Moteur associatif Qlik. Vous pouvez spécifier une ou plusieurs colonnes. Les valeurs manquantes sont automatiquement renseignées par Moteur associatif Qlik.

Syntaxe :

```
Add loadstatement
```

Exemple :

Cet exemple ajoute deux lignes aux colonnes *Dates* et *Sales* de l'instruction inline.

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

Dans un contexte de modification de graphique, le préfixe **Replace** remplace toutes les valeurs de la table *HC1* par une valeur calculée définie par le script.

Syntaxe :

```
Replace loadstatement
```

Exemple :

Cet exemple remplace toutes les valeurs de la colonne `z` par la somme de `x` et de `y`.

```
Replace Load
x+y as z
Resident HC1;
```

6.6 Instructions normales

Les instructions normales servent généralement à manipuler des données d'une manière ou d'une autre. Ces instructions peuvent être écrites sur autant de lignes de script que nécessaire et doivent toujours se terminer par un point-virgule « ; ».

Tous les mots-clés du script peuvent être saisis en majuscules et/ou en minuscules. Les noms des champs et des variables utilisés dans les instructions sont toutefois sensibles à la casse des caractères.

Vue d'ensemble des instructions normales d'un modificateur de graphique

Chaque fonction est décrite plus en détail après la vue d'ensemble. Vous pouvez également cliquer sur le nom de la fonction qui vous intéresse dans la syntaxe afin d'accéder immédiatement aux informations connexes.

LOAD

Dans un contexte de modification de graphique, l'instruction **LOAD** charge des données supplémentaires dans l'hypercube à partir des données définies dans le script ou d'une table précédemment chargée. Il est également possible de charger des données à partir de connexions analytiques.

*L'instruction **LOAD** doit avoir le préfixe **Replace** ou **Add**, ou elle sera rejetée.*

```
Add | Replace Load [ distinct ] fieldlist
(
inline data [ format-spec ] |
resident table-label
) | extension pluginname.functionname([script] tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Let

L'instruction **let** complète l'instruction **set**, utilisée pour définir des variables de script. Contrairement à l'instruction **set**, l'instruction **let** évalue l'expression située à droite du signe '=' lors de l'exécution du script avant qu'elle soit attribuée à la variable.

```
Let variablename=expression
```

Set

L'instruction **set** permet de définir des variables de script. Ces variables peuvent servir à remplacer des chaînes, des chemins d'accès, des lecteurs, etc.

```
Set variablename=string
```

Put

L'instruction **Put** permet de définir une valeur numérique dans l'hypercube.

HCValue

L'instruction **HCValue** permet de récupérer les valeurs d'une ligne d'une colonne spécifiée.

Load

Dans un contexte de modification de graphique, l'instruction **LOAD** charge des données supplémentaires dans l'hypercube à partir des données définies dans le script ou d'une table précédemment chargée. Il est également possible de charger des données à partir de connexions analytiques.

*L'instruction **LOAD** doit avoir le préfixe **Replace** ou **Add**, ou elle sera rejetée.*

Syntaxe :

```
Add | Replace LOAD fieldlist
(
inline data [ format-spec ] |
resident table-label
) | extension pluginname.functionname([script] tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Arguments :

Arguments

Argument	Description
fieldlist	*fieldlist ::= (* | field {, * | field })* Liste des champs à charger. L'utilisation du symbole * comme liste de champs signifie inclure tous les champs de la table. *field ::= (fieldref | expression) [***as** *aliasname]* La définition du champ doit toujours contenir un littéral, une référence à un champ existant ou une expression. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* est un texte identique à un nom de champ dans la table. Notez que le nom du champ doit être mis entre guillemets doubles droits ou entre crochets s'il contient des espaces, par exemple. Les noms des champs ne sont pas toujours disponibles de manière explicite. Une notation différente est alors utilisée : @*fieldnumber* représente le numéro du champ dans un fichier de table délimité. Il doit s'agir d'un entier positif précédé d'un arobase (@). La numérotation est toujours effectuée de 1 jusqu'au nombre total de champs. @*startpos*:*endpos* représente les positions de début et de fin d'un champ dans un fichier contenant des enregistrements de longueur fixe. Ces positions doivent être toutes deux des entiers positifs. Les deux nombres doivent être précédés d'un arobase (@) et séparés par deux-points. La numérotation est toujours effectuée de 1 jusqu'au nombre total de positions. Dans le dernier champ, **n** est utilisé comme position de fin. • Si @*startpos*:*endpos* est immédiatement suivi des caractères **I** ou **U**, les octets lus seront interprétés comme un entier binaire signé (**I**) ou non signé (**U**) (selon l'ordre des octets d'Intel). Le nombre de positions lues doit être égal à 1, 2 ou 4. • Si @*startpos*:*endpos* est immédiatement suivi du caractère **R**, les octets lus seront interprétés comme un nombre réel binaire (en virgule flottante de 32 bits ou 64 bits IEEE). Le nombre de positions lues doit être égal à 4 ou 8. • Si @*startpos*:*endpos* est immédiatement suivi du caractère **B**, les octets lus seront interprétés comme des nombres BCD (Binary Coded Decimal) selon la norme COMP-3. Vous pouvez spécifier n'importe quel nombre d'octets. *expression* peut correspondre à une fonction numérique ou une fonction de chaîne basée sur un ou plusieurs autres champs de la même table. Pour plus d'informations, voir la syntaxe des expressions. **as** est utilisé pour attribuer un nouveau nom au champ.

Argument	Description
inline	**inline** est utilisé si les données doivent être saisies dans le script au lieu d'être chargées à partir d'un fichier. *data ::= [text]* Les données saisies à l'aide d'une clause **inline** doivent être mises entre guillemets ou entre crochets. Le texte placé entre ces guillemets ou crochets est interprété de la même manière que le contenu d'un fichier. C'est pourquoi vous devez également insérer une nouvelle ligne dans la clause **inline**, là où vous en auriez inséré une dans un fichier texte. Pour ce faire, appuyez sur la touche Entrée lors de la saisie du script. Le nombre de colonnes est défini dans la première ligne. *format-spec ::= (fspec-item {, fspec-item })* La spécification du format se compose d'une liste de plusieurs éléments de spécification du format, mis entre parenthèses.
resident	**resident** est utilisé si les données doivent être chargées à partir d'une table précédemment chargée. *table label* est une étiquette précédant l'instruction **LOAD** ayant créé la table de départ. L'étiquette doit être saisie avec deux-points à la fin.

Argument	Description
extension	Vous pouvez charger des données à partir de connexions analytiques. Vous devez utiliser la clause **extension** pour appeler une fonction définie dans le plug-in SSE (Server-Side Extension) ou pour évaluer un script. Il est possible d'envoyer une seule table au plug-in SSE ; une seule table de données est alors renvoyée. Si le plug-in ne précise pas les noms des champs renvoyés, les champs seront nommés Field1, Field2, et ainsi de suite. `Extension pluginname.functionname( tabledescription );` • Chargement de données à l'aide d'une fonction dans un plug-in SSE *tabledescription ::= (table { ,tablefield})* Si vous ne déclarez pas les champs de table, ils sont utilisés dans l'ordre de chargement. • Chargement de données via l'évaluation d'un script dans un plug-in SSE *tabledescription ::= (script, table { ,tablefield})* **Gestion des types de données dans la définition des champs de table** Les types de données sont automatiquement détectés dans les connexions analytiques. Si les données ne comportent aucune valeur numérique et comprennent au moins une chaîne de texte non NULLE, le champ est interprété comme du texte. Dans tous les autres cas, il est considéré comme de type numérique. Vous pouvez appliquer un type de données forcé en encadrant le nom d'un champ à l'aide de **String()** ou de **Mixed()**. • **String()** convertit de force le champ en texte. Si le champ est numérique, la partie texte de la valeur double est extraite et aucune conversion n'est effectuée. • **Mixed()** convertit de force le champ en valeur double. Il est impossible d'utiliser **String()** ou **Mixed()** en dehors des définitions de champ de table **extension**. De plus, vous ne pouvez pas utiliser d'autres fonctions Qlik Sense dans une définition de champ de table.
where	**where** est une clause utilisée pour indiquer si un enregistrement doit être inclus ou pas dans la sélection. La sélection est incluse si l'expression *criterion* est définie sur True. *criterion* est une expression logique.
while	**while** est une clause utilisée pour indiquer si un enregistrement doit être lu plusieurs fois. Le même enregistrement est lu tant que l'expression *criterion* est définie sur True. Pour être utile, une clause **while** doit généralement inclure la fonction **IterNo( )**. *criterion* est une expression logique.

Argument	Description
group by	**group by** est une clause utilisée pour déterminer les champs sur lesquels les données doivent être agrégées (groupées). Les champs d'agrégation doivent être inclus d'une manière ou d'une autre dans les expressions chargées. Aucun autre champ que les champs d'agrégation ne peut être utilisé en dehors des fonctions d'agrégation dans les expressions chargées. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** est une clause utilisée pour trier les enregistrements d'une table résidente avant qu'ils ne soient traités par l'instruction **load**. La table résidente peut être triée par un ou plusieurs champs, par ordre croissant ou décroissant. Le tri est principalement effectué par valeur numérique, puis par valeur de paramètres régionaux de classement national. Cette clause peut uniquement être utilisée lorsque la source de données est une table résidente. Les champs de tri indiquent les champs selon lesquels la table résidente est triée. Le champ peut être spécifié par son nom ou par son numéro dans la table résidente (le premier champ est le numéro 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* correspond soit à *asc* pour un ordre croissant, soit à *desc* pour un ordre décroissant. Si aucun argument *sortorder* n'est spécifié, c'est *asc* qui est utilisé. *fieldname*, *path*, *filename* et *aliasname* sont des chaînes textuelles représentant ce que ces noms désignent. N'importe quel champ de la table source peut être utilisé comme *fieldname*. Toutefois, les champs créés via la clause as (*aliasname*) ne sont pas concernés et ne peuvent pas être utilisés dans la même instruction **load**.

Let

L'instruction **let** complète l'instruction **set**, utilisée pour définir des variables de script. Contrairement à l'instruction **set**, l'instruction **let** évalue l'expression située à droite du signe '=' lors de l'exécution du script avant qu'elle soit attribuée à la variable.

Syntaxe :

```
Let variablename=expression
```

Exemples et résultats :

Exemple	Résultat

`Set x=3+4;`
`Let y=3+4;`
`z=$(y)+1;`
	`$(x)` est évalué en tant que ' 3+4'.
	`$(y)` est évalué en tant que ' 7'.
	`$(z)` est évalué en tant que ' 8'.
	Notez la différence entre les instructions **Set** et **Let**. L'instruction **Set** assigne la chaîne '3+4' à la variable, tandis que l'instruction **Let** évalue la chaîne et assigne 7 à la variable.

Let T=now( );	`$(T)` prend la valeur de l'heure active.

Set

L'instruction **set** permet de définir des variables de script. Ces variables peuvent servir à remplacer des chaînes, des chemins d'accès, des lecteurs, etc.

Syntaxe :

```
Set variablename=string
```

Example 1:

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

Example 2:

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

Example 3:

```
Set BudgetYear=2012;
```

Put

L'instruction **put** permet de définir une valeur numérique dans l'hypercube.

L'accès aux colonnes peut s'effectuer par les étiquettes. Vous pouvez également accéder aux colonnes et aux lignes par ordre de déclaration. Pour plus de détails, voir les exemples ci-dessous.

Syntaxe :

```
put column(position)=value
```

Example 1:

L'accès aux colonnes peut s'effectuer par les étiquettes.

Cet exemple définit une valeur de `1` à la première position de la colonne étiquetée *Sales*.

```
Put Sales(1) = 1;
```

Example 2:

Vous pouvez accéder aux colonnes de mesures par ordre de déclaration via le format `#hc1.measure` pour les mesures.

Cet exemple définira la valeur 1 000 à la dixième position de l'hypercube trié final.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Vous pouvez accéder aux lignes de dimensions par ordre de déclaration via le format `#hc1.dimension` pour les dimensions.

Cet exemple place la valeur de la constante Pi à la cinquième ligne de la troisième dimension déclarée.

```
Put #hc1.dimension.3(5) = Pi();
```

En l'absence desdites dimensions ou expressions, dans la valeur ou sur les étiquettes, une erreur est renvoyée, indiquant que la colonne n'a pas été trouvée. Si l'index de la colonne est hors limites, aucune erreur n'est affichée.

HCValue

La fonction **HCValue** permet de récupérer les valeurs d'une ligne d'une colonne spécifiée.

Syntaxe :

```
HCValue(column,position)
```

Example 1:

Cet exemple renvoie la valeur à la première position de la colonne portant l'étiquette ‘Sales’.

```
HCValue(Sales,1)
```

Example 2:

Cet exemple renvoie la valeur à la dixième position de l'hypercube trié.

```
HCValue(#hc1.measure.2,10)
```

Example 3:

Cet exemple renvoie la valeur à la cinquième ligne de la troisième dimension.

```
HCValue(#hc1.dimension.3,5)
```

En l'absence desdites dimensions ou expressions, en valeur ou étiquettes, une erreur est renvoyée, indiquant que la colonne n'a pas été trouvée. Si l'index de la colonne est hors limites, la valeur NULL est renvoyée.

7 Fonctions et instructions QlikView non prises en charge dans Qlik Sense

La plupart des fonctions et instructions qui s'emploient dans les scripts de chargement et les expressions de graphique QlikView sont également prises en charge dans Qlik Sense, mais il existe des exceptions, comme décrit dans cette section.

7.1 Instructions de script non prises en charge dans Qlik Sense

Instructions de script QlikView non prises en charge dans Qlik Sense

Instruction	Commentaires
Command	Utilisez **SQL** à la place.
InputField	

7.2 Fonctions non prises en charge dans Qlik Sense

La liste suivante décrit les fonctions de script et de graphique QlikView qui ne sont pas prises en charge dans Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Préfixes non pris en charge dans Qlik Sense

La liste suivante décrit les préfixes QlikView qui ne sont pas pris en charge dans Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

8 Fonctions et instructions déconseillées dans Qlik Sense

La plupart des fonctions et instructions qui s'emploient dans les scripts de chargement et les expressions de graphique QlikView sont également prises en charge dans Qlik Sense, mais certaines d'entre elles sont déconseillées dans Qlik Sense. De plus, certaines fonctions et instructions disponibles dans les versions précédentes de Qlik Sense ont été dépréciées.

Pour des raisons de compatibilité, elles fonctionneront comme prévu. Il est toutefois conseillé de mettre à jour le code d'après les recommandations de cette section, car ces fonctions et instructions risquent d'être retirées des prochaines versions.

8.1 Instructions de script déconseillées dans Qlik Sense

La table suivante contient les instructions de script qu'il est déconseillé d'utiliser dans Qlik Sense.

Instructions de script déconseillées

Instruction	Recommandation
Command	Utilisez **SQL** à la place.
CustomConnect	Utilisez **Custom Connect** à la place.

8.2 Paramètres d'instruction de script déconseillés dans Qlik Sense

La table suivante décrit les paramètres d'instruction de script qu'il est déconseillé d'utiliser dans Qlik Sense.

Paramètres d'instruction de script déconseillés

Instruction	Paramètres
Buffer	Utilisez **Incremental** à la place des paramètres : • **Inc** (déconseillé) • **Incr** (déconseillé)

Instruction	Paramètres
LOAD	Les mots-clés de paramètre suivants sont générés par les assistants de transformation de fichier QlikView. Les fonctionnalités sont conservées lors du rechargement des données, mais Qlik Sense ne fournit pas de prise en charge ou d'assistants pour générer l'instruction avec ces paramètres : • Bottom • Cellvalue • Col • Colmatch • Colsplit • Colxtr • Compound • Contain • Equal • Every • Expand • Filters • Intarray • Interpret • Length • Longer • Numerical • Pos • Remove • Rotate • Row • Rowcnd • Shorter • Start • Strcnd • Top • Transpose • Unwrap • XML: XMLSAX and Pattern is Path

8.3 Fonctions déconseillées dans Qlik Sense

La table suivante décrit les fonctions de script et de graphique qu'il est déconseillé d'utiliser dans Qlik Sense.

Fonctions déconseillées

Fonction	Recommandation
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Utilisez les fonctions de plage à la place. *Fonctions de plage (page 1333)*
Color() **QliktechBlue** **QliktechGray**	Utilisez d'autres fonctions de couleur à la place. Il est possible de remplacer **QliktechBlue()** par **RGB(8, 18, 90)** et **QliktechGray** par **RGB(158, 148, 137)** pour obtenir les mêmes couleurs. *Fonctions de couleur (page 551)*
QlikViewVersion	Utilisez **EngineVersion** à la place. *EngineVersion (page 1465)*
ProductVersion	Utilisez **EngineVersion** à la place. *EngineVersion (page 1465)*
QVUser	
Year2Date	Utilisez **YearToDate** à la place.
Vrank	Utilisez **Rank** à la place.
WildMatch5	Utilisez **WildMatch** à la place.

Qualificateur ALL

Dans QlikView, le qualificateur **ALL** peut précéder une expression. Cela équivaut à utiliser **{1} TOTAL**. Dans un tel cas, le calcul est effectué sur toutes les valeurs du champ figurant dans le document, mais les dimensions du graphique et les sélections actives sont ignorées. La même valeur est renvoyée systématiquement, quel que soit l'état logique du document. Si le qualificateur **ALL** est utilisé, il n'est pas possible d'employer une expression d'ensemble, puisque le qualificateur **ALL** définit un ensemble. Pour des raisons de compatibilité entre versions, le qualificateur **ALL** fonctionne toujours dans cette version de Qlik Sense, mais il risque d'être retiré des versions ultérieures.