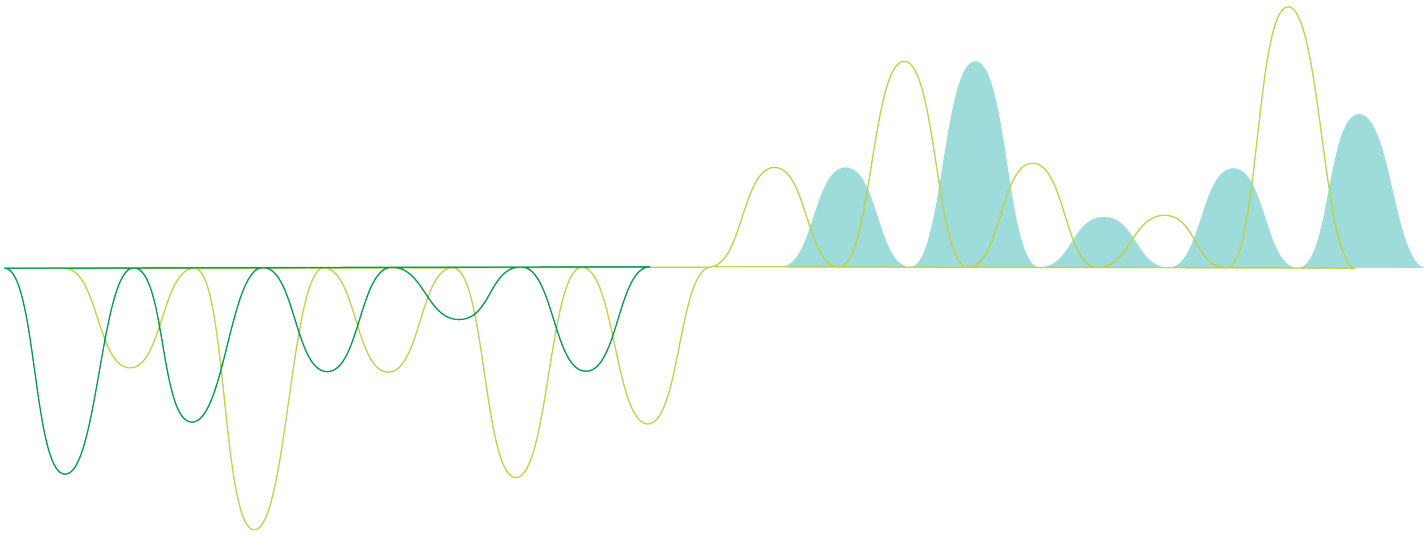
Kod söz dizimi ve grafik fonksiyonları

Qlik Sense®
May 2022



Contents

Contents

1 Qlik Sense nedir?

Qlik Sense, veri analizi için bir platformdur. Qlik Sense ile verileri analiz edebilir ve kendi başınıza veri keşifleri yapabilirsiniz. Gruplarda ve kuruluşlar arasında bilgileri paylaşabilir ve verileri analiz edebilirsiniz. Qlik Sense, kendi sorularınızı sormanıza ve olayları kavrarken kendi yolunuzu izlemenize olanak sağlar. Qlik Sense sizin ve iş arkadaşlarınızın iş birliği içinde karar vermelerine olanak sağlar.

1.1 Qlik Sense uygulamasında neler yapabilirsiniz?

Çoğu İş Zekası (BI) ürünü, anlaşılan soruları önceden yanıtlamanıza yardımcı olabilir. Peki takip eden sorularınız ne olacak? Birisi raporunuzu okuduktan ve görselleştirmenizi gördükten sonra gelenler? Qlik Sense ilişkisel deneyimiyle, bilgiye giden kendi yolunuzda ilerleyerek arka arkaya sorular yanıtlayabilirsiniz. Qlik Sense ile, verilerinizi yalnızca birkaç tıklamayla araştırabilir, her adımda yeni şeyler öğrenebilir ve daha önce keşfedilenlere göre sonraki adımları belirleyebilirsiniz.

1.2 Qlik Sense nasıl çalışır?

Qlik Sense, sizin için anında bilgi görünümleri oluşturur. Qlik Sense, önceden tanımlanmış veya statik raporlar ya da diğer kullanıcılara bağımlı olmanızı gerektirmez; öğrenmek için tıklamanız yeterlidir. Her tıklayışınızda, Qlik Sense uygulamadaki her Qlik Sense görselleştirmesini ve görünümünü seçimlerinize özgü yeni hesaplanan veri ve görselleştirme kümesiyle güncelleyerek anında yanıt verir.

Uygulama modeli

Büyük iş uygulamaları dağıtmak ve yönetmek yerine, yeniden kullanabileceğiniz, değiştirebileceğiniz ve başkalarıyla paylaşabileceğiniz kendi Qlik Sense uygulamalarınızı oluşturabilirsiniz. Uygulama modeli, yeni rapor veya görselleştirme için bir uzmana dönmek zorunda kalmadan bir sonraki soruyu kendi başınıza sormanıza ve yanıtlamanıza yardımcı olur.

İlişkisel deneyim

Qlik Sense, verilerdeki tüm ilişkileri otomatik olarak yönetir ve **green/white/gray** metaforu kullanarak size bilgi sunar. Seçimler yeşil olarak vurgulanır, ilişkili veriler beyazla temsil edilir ve hariç tutulan (ilişkilendirilmemiş) veriler gri olarak görünür. Bu anında geri bildirim, sonraki soruları düşünmenizi ve araştırmaya ve keşfetmeye devam etmenizi sağlar.

İşbirliği ve hareketlilik

Qlik Sense, ne zaman isterseniz iş arkadaşlarınızla istediğiniz yerden işbirliği yapmanızı sağlar. İlişkisel deneyim ve iş birliği dahil tüm Qlik Sense özellikleri mobil cihazlarda kullanılabilir. Qlik Sense ile, nerede olursanız olun iş arkadaşlarınızla birlikte sorularınızı ve takip eden sorularınızı sorabilirsiniz.

1.3 Qlik Sense uygulamasını nasıl dağıtabilirsiniz?

Qlik Sense uygulamasının dağıtılabilecek iki sürümü vardır: Qlik Sense Desktop ve Qlik Sense Enterprise.

Qlik Sense Desktop

Bu, yüklenmesi kolay olan ve genellikle yerel bilgisayara yüklenen tek kullanıcı sürümüdür.

Qlik Sense Enterprise

Bu sürüm, Qlik Sense sitelerini dağıtmak için kullanılır. Bir site, ortak mantıksal depo veya merkezi düğüme bağlı bir veya daha fazla sunucu makinesinden oluşan bir koleksiyondur.

1.4 Qlik Sense sitesinin yönetimi

Qlik Management Console ile, Qlik Sense sitelerini kolay ve sezgisel bir şekilde yapılandırabilir, yönetebilir ve izleyebilirsiniz. Lisansları, erişim ve güvenlik kurallarını yönetebilir, düğümleri ve veri kaynağı bağlantılarını yapılandırabilir ve içeriği ve kullanıcıları diğer birçok etkinlik ve kaynak arasında eşzamanlayabilirsiniz.

1.5 Qlik Sense uygulamasını geliştirme ve kendi amaçlarınıza uyarlama

Qlik Sense, kendi uzantılarınızı geliştirmek ve Qlik Sense uygulamasını şunlar gibi farklı amaçlar için uyarlamak ve tümleştirmek için size esnek API'ler ve SDK'lar sunar:

Uzantılar ve karma ortamlar oluşturma

Burada, Qlik Sense uygulamalarında özel görselleştirme olan uzantılar oluşturmak için JavaScript'i kullanarak web geliştirme gerçekleştirebilir veya Qlik Sense içeriğiyle web siteleri oluşturmak için karma ortam API'lerini kullanabilirsiniz.

İstemci oluşturma

.NET'te istemciler oluşturabilir ve Qlik Sense nesnelerini kendi uygulamalarınıza ekleyebilirsiniz. Qlik Sense istemci protokolünü kullanarak WebSocket iletişimini yönetebilecek herhangi bir programlama dilinde yerel istemciler de oluşturabilirsiniz.

Sunucu araçları oluşturma

Hizmet ve kullanıcı dizini API'leriyle, Qlik Sense sitelerini yönetmek için kendi aracınızı oluşturabilirsiniz.

Diğer veri kaynaklarına bağlanma

Özel veri kaynaklarından veri almak için Qlik Sense bağlayıcıları oluşturun.

2 Kod söz dizimine genel bakış

2.1 Kod söz dizimine giriş

Kod içinde, mantığa dahil edilen veri kaynağının adı, tabloların adları ve alanların adları tanımlanır. Buna ek olarak, erişim hakları tanımındaki alanlar da kod içinde tanımlanır. Kod, art arda yürütülen bir dizi deyimden oluşur.

Qlik Sense komut satırı söz dizimi ve kod söz dizimi, Backus-Naur Biçimciliği (veya BNF) olarak adlandırılan bir gösterimde açıklanır.

Yeni bir Qlik Sense dosyası oluşturulduğunda kodun ilk satırları zaten oluşturulmuş olur. Bu sayı yorumlama değişkenlerinin varsayılan değerleri işletim sisteminin bölge ayarlarından türetilir.

Kod, art arda yürütülen bir dizi kod deyiminden ve anahtar sözcüklerden oluşur. Tüm kod deyimleri bir noktalı virgül **";"** işaretiyle sonlanmalıdır.

Yüklenen verileri dönüştürmek için **LOAD** deyimlerindeki ifadeleri ve fonksiyonları kullanabilirsiniz.

Sınırlayıcı olarak virgül, sekme veya noktalı virgüllerin bulunduğu bir tablo dosyası için **LOAD** deyimi kullanılabilir. Varsayılan olarak, **LOAD** deyimi dosyanın tüm alanlarını yükler.

ODBC veya OLE DBveritabanı bağlayıcılarıyla genel veritabanlarına erişilebilir. Burada, standart SQL deyimleri kullanılır. Kabul edilen SQL söz dizimi farklı ODBC sürücüleri arasında değişiklik gösterir.

Ayrıca, özel bağlayıcıları kullanarak diğer veri kaynaklarına erişebilirsiniz.

2.2 Backus-Naur formalizmi nedir?

Qlik Sense komut satırı söz dizimi ve kod söz dizimi, Backus-Naur biçimciliği olarak adlandırılan (BNF kodu olarak da bilinir) bir gösterimde açıklanır.

Aşağıdaki tabloda, BNF kodunda kullanılan sembollerin bir listesi ile birlikte, bunların nasıl yorumlandığıyla ilgili bir açıklama verilmektedir:

Simgeler

Sembol	Açıklama
|	Mantıksal OR: Her iki taraftaki sembol kullanılabilir.
( )	Önceliği tanımlayan parantezler: BNF söz dizimini yapılandırmak için kullanılır.
[ ]	Köşeli ayraçlar: içindeki öğeler isteğe bağlıdır.
{ }	Kaşlı ayraçlar: içindeki öğeler sıfır veya daha fazla sayıda yinelenebilir.
Sembol	Terminal olmayan söz dizimsel kategori: Daha başka sembollere bölünebilir. Örneğin, yukarıdakilerin bileşimleri, diğer terminal olmayan semboller, metin dizeleri vs.
::=	Sembolü tanımlayan blokun başlangıcını belirtir.
LOAD	Bir metin dizesinden oluşan terminal sembolü. Koda olduğu gibi yazılmalıdır.

Tüm terminal semboller **bold face** yazı tipiyle yazılır. Örneğin; "(" önceliği belirleyen bir parantez olarak yorumlanması gerekirken, "**(**" koda yazılacak bir karakter olarak yorumlanmalıdır.

Örnek:

Alias deyiminin tanımı şöyledir:

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

Bu, "alias" metin dizesi, ardından isteğe bağlı alan adı, ardından "as" metin dizesi, ardından isteğe bağlı alias adı olarak yorumlanmalıdır. İstenilen sayıda "fieldname as alias" ek kombinasyonu virgülle ayrılmış olarak verilebilir.

Aşağıdakiler doğru deyimlerdir:

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

Aşağıdaki deyimler doğru değildir:

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

2 Kod deyimleri ve anahtar sözcükler

Qlik Sense kodu bir dizi deyimden oluşur. Deyimler, normal bir kod deyimi veya bir kod kontrol ifadesi olabilir. Belirli deyimlerden önce önekler gelebilir.

Normal deyimler genellikle verileri birkaç farklı şekilde işlemek için kullanılır. Bu deyimler kod içinde birçok satıra yazılabilir ve her zaman bir noktalı virgül ";" işaretiyle sonlandırılmalıdır.

Kontrol ifadeleri genellikle kod yürütme akışını kontrol etmek için kullanılır. Bir kontrol ifadesinin her bir cümlesi, bir kod satırı içinde tutulmalı ve noktalı virgül veya satır sonu ile sonlandırılmalıdır.

Önekler uygulanabilir durumdaki normal deyimlere uygulanabilir; ancak kontrol ifadelerine asla uygulanamaz. Bununla birlikte **when** ve **unless** önekleri birkaç belirli kontrol ifadesi cümlesinde sonek olarak kullanılabilir.

Bir sonraki alt bölümde tüm kod deyimlerinin, kontrol ifadelerinin ve öneklerin alfabetik bir listesi bulunmaktadır.

Tüm kod anahtar sözcükleri küçük harf ve büyük harften oluşan karakterlerin herhangi bir bileşimiyle yazılabilir. Bununla birlikte, deyimlerde kullanılan alan ve değişken adları büyük/küçük harf duyarlıdır.

2.3 Kod kontrol ifadeleri

Qlik Sense kodu bir dizi deyimden oluşur. Deyimler, normal bir kod deyimi veya bir kod kontrol ifadesi olabilir.

Kontrol ifadeleri genellikle kod yürütme akışını kontrol etmek için kullanılır. Bir kontrol ifadesinin her bir cümlesi, bir kod satırı içinde tutulmalı ve noktalı virgül veya satır sonu ile sonlandırılmalıdır.

Birkaç belirli kontrol ifadesiyle kullanılabilen **when** ve **unless** önekleri istisna olmak üzere, önekler kontrol ifadelerinde asla uygulanmaz.

Tüm kod anahtar sözcükleri küçük harf ve büyük harften oluşan karakterlerin herhangi bir bileşimiyle yazılabilir.

Kod kontrol ifadelerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Call

call kontrol ifadesi, önceki bir **sub** deyimiyle tanımlanmış olması gereken bir alt rutini çağırır.

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

Do..loop

do..loop kontrol ifadesi, mantıksal koşul sağlanıncaya kadar bir veya daha fazla deyimi yürüten bir kod yineleme yapısıdır.

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

Exit script

Kontrol ifadesi kod yürütmeyi durdurur. Kodda herhangi bir yere eklenebilir.

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

For each ..next

for each..next kontrol ifadesi, virgülle ayrılmış listedeki her bir değer için bir veya daha fazla deyimi yürüten bir kod yineleme yapısıdır. **for** ve **next** öğeleri arasına alınan döngüdeki deyimler, listedeki her bir değer için yürütülür.

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

For..next

for..next kontrol ifadesi, sayaçlı bir kod yineleme yapısıdır. **for** ve **next** öğelerinin içine aldığı döngünün içindeki deyimler, belirtilen düşük ve yüksek sınırlar arasındaki sayaç değişkeninin her bir değeri için yürütülür.

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

If..then

if..then kontrol ifadesi, bir veya daha fazla mantıksal koşula bağlı olarak farklı yolları takip etmesi için kod yürütmesini zorlayan komut seçim yapısıdır.

> ***if..then*** *deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, bu deyimin dört olası cümlesinin her biri (**if..then**, **elseif..then**, **else** ve **end if**) satır sınırını geçmemelidir.*

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

Sub

sub..end sub kontrol ifadesi, bir **call** deyimiyle çağrılabilecek bir alt yordam tanımlar.

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

Switch

switch kontrol ifadesi, ifade değerine bağlı olarak, yolları takip etmek için kod yürütmesini zorlayan bir kod seçim yapısıdır.

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

Call

call kontrol ifadesi, önceki bir **sub** deyimiyle tanımlanmış olması gereken bir alt rutini çağırır.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
name	Alt rutinin adı.
paramlist	Alt rutine gönderilecek olan gerçek parametrelerin virgülle ayrılmış listesi. Listedeki her öğe bir alan adı, değişken veya rastgele seçilmiş bir ifade olabilir.

Bir **call** deyimiyle çağrılan alt rutin, kod yürütme sırasında daha önce karşılaşılan bir **sub** ile tanımlanmış olmalıdır.

Parametreler alt rutine kopyalanır ve **call** deyimindeki parametre bir değişkense ve bir ifade değilse, alt rutinden çıktıktan sonra tekrar dışarı kopyalanır.

Sınırlamalar:

- **call** deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgül veya satır sonu ile bittiğinden, satır sınırını geçmemelidir.
- Bir kontrol ifadesinde `Sub..end sub` ile bir alt rutin tanımladığınızda, ör. `if..then`, alt rutini yalnızca aynı kontrol ifadesinden çağırabilirsiniz.

Örnek:

Bu örnek, klasördeki ve alt klasörlerindeki Qlik ile ilgili tüm dosyaları listeler ve dosya bilgilerini bir tabloda depolar. Klasörde Apps adlı bir veri bağlantısı oluşturduğunuz varsayılır.

DoDir alt rutini, parametre olarak 'lib://Apps' klasörüne yapılan bir referansla çağrılır. Alt rutinin içerisinde, fonksiyonun alt klasörlerde yinelemeli olarak dosya aramasını sağlayan yinelemeli `Call DoDir (Dir)` çağrısı bulunur.

```
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

do..loop kontrol ifadesi, mantıksal koşul sağlanıncaya kadar bir veya daha fazla deyimi yürüten bir kod yineleme yapısıdır.

Söz Dizimi:

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

> ***do..loop*** *deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, üç olası cümlesinin her biri (**do**, **exit do** ve **loop**) satır sınırını geçmemelidir.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.
while / until	**while** veya **until** koşullu cümleleri herhangi bir **do..loop** deyimi içinde yalnızca bir kez görünmelidir; yani ya **do** öğesinden sonra ya da **loop** öğesinden sonra görünmelidir. Her bir koşul yalnızca karşılaşıldığı ilk seferde yorumlanır, ancak döngü içinde karşılaşıldığı her seferinde değerlendirilir.
exit do	Döngü içinde bir **exit do** cümlesiyle karşılaşılırsa, kodun yürütülmesi döngünün sonunu belirten **loop** cümlesinden sonra gelen ilk deyime aktarılır. Bir **exit do** cümlesi, **when** veya **unless** sonekinin isteğe bağlı kullanımıyla koşullu hale getirilebilir.

Örnek:

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

End

End kod anahtar sözcüğü **If**, **Sub** ve **Switch** cümlelerini kapatmak için kullanılır.

Exit

Exit kod anahtar sözcüğü **Exit Script** deyiminin bir parçasıdır; ancak **Do**, **For** veya **Sub** cümlelerinden çıkmak için de kullanılabilir.

Exit script

Kontrol ifadesi kod yürütmeyi durdurur. Kodda herhangi bir yere eklenebilir.

Söz Dizimi:

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

exit script deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgül veya satır sonu ile bittiğinden, satır sınırını geçmemelidir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
when / unless	Bir **exit script** deyimi, **when** veya **unless** cümlesinin isteğe bağlı kullanımıyla koşullu hale getirilebilir.

Örnekler:

```
//Exit script
Exit Script;

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

For..next

for..next kontrol ifadesi, sayaçlı bir kod yineleme yapısıdır. **for** ve **next** öğelerinin içine aldığı döngünün içindeki deyimler, belirtilen düşük ve yüksek sınırlar arasındaki sayaç değişkeninin her bir değeri için yürütülür.

Söz Dizimi:

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

expr1, *expr2* ve *expr3* ifadeleri yalnızca döngüye ilk girildiğinde değerlendirilir. Counter değişkeninin değeri döngü içinde deyimlerle değiştirilebilir, ancak bu iyi bir programlama uygulaması değildir.

Döngü içinde bir **exit for** cümlesiyle karşılaşılırsa, kodun yürütülmesi döngünün sonunu belirten **next** cümlesinden sonra gelen ilk deyime aktarılır. Bir **exit for** cümlesi, **when** veya **unless** sonekinin isteğe bağlı kullanımıyla koşullu hale getirilebilir.

> ***for..next*** *deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, üç olası cümlesinin her biri (****for..to..step****,* ***exit for*** *ve* ***next****) satır sınırını geçmemelidir.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
counter	Bir değişken adı. *counter* öğesi **next** öğesinden sonra belirtilirse, karşılık gelen **for** öğesinden sonra bulunan öğeyle aynı değişken adı olmalıdır.
expr1	Döngünün yürütülmesi gereken *counter* değişkeninin ilk değerini belirleyen bir ifade.
expr2	Döngünün yürütülmesi gereken *counter* değişkeninin son değerini belirleyen bir ifade.
expr3	Döngü her yürütüldüğünde *counter* değişkeninin artımını gösteren değeri belirleyen bir ifade.
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.

Example 1: Bir dosya dizisi yükleme

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

      LOAD * from file$(a).csv;

next
```

Example 2: Rastgele sayıda dosya yükleme

Bu örnekte, *x1.csv, x3.csv, x5.csv, x7.csv* ve *x9.csv* veri dosyaları olduğunu varsayıyoruz. `if rand( )<0.5 then` koşulu kullanılarak, yükleme rastgele bir noktada durdurulur.

```
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

for each..next kontrol ifadesi, virgülle ayrılmış listedeki her bir değer için bir veya daha fazla deyimi yürüten bir kod yineleme yapısıdır. **for** ve **next** öğeleri arasına alınan döngüdeki deyimler, listedeki her bir değer için yürütülür.

Söz Dizimi:
Özel söz dizimi geçerli dizinde dosya ve dizin adlarıyla listeler oluşturmayı mümkün kılar.

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
var	Her bir döngü yürütmesi için listeden yeni bir değer edinecek kod değişkeni adı. **var** öğesi **next** öğesinden sonra belirtilirse, karşılık gelen **for each** öğesinden sonra bulunan öğeyle aynı değişken adı olmalıdır.

var değişkeninin değeri döngü içinde deyimlerle değiştirilebilir, ancak bu iyi bir programlama uygulaması değildir.

Döngü içinde bir **exit for** cümlesiyle karşılaşılırsa, kodun yürütülmesi döngünün sonunu belirten **next** cümlesinden sonra gelen ilk deyime aktarılır. Bir **exit for** cümlesi, **when** veya **unless** sonekinin isteğe bağlı kullanımıyla koşullu hale getirilebilir.

> ***for each..next*** *deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, üç olası cümlesinin her biri (****for each****, ****exit for*** *ve* ***next****) satır sınırını geçmemelidir.*

Söz Dizimi:

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

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
constant	Herhangi bir sayı veya dize. Doğrudan koda yazılan bir dizenin tek tırnak içine alınması gerektiğini unutmayın. Tek tırnak içinde olmayan dize bir değişken olarak yorumlanır ve değişkenin değeri kullanılır. Sayıların tek tırnak içine alınması gerekmez.
expression	Rastgele seçilen bir ifade.
mask	Geçerli dosya adı karakterlerini ve aynı zamanda standart joker karakterlerini (***** ve **?**) de içerebilen bir dosya adı veya klasör adı maskesi. Mutlak dosya yollarını veya lib:// yollarını kullanabilirsiniz.
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.
filelist mask	Bu söz dizimi, geçerli dizinde olup dosya adı maskesiyle eşleşen tüm dosyaların virgülle ayrılmış bir listesini oluşturur. *Bu bağımsız değişken, yalnızca standart modda kütüphane bağlantılarını destekler.*
dirlist mask	Bu söz dizimi, geçerli klasörde olup klasör adı maskesiyle eşleşen tüm klasörlerin virgülle ayrılmış bir listesini oluşturur. *Bu bağımsız değişken, yalnızca standart modda kütüphane bağlantılarını destekler.*
fieldvaluelist mask	Bu söz dizimi, Qlik Sense içine önceden yüklenmiş bir alanın değerleri aracılığıyla yinelenir.

*Qlik Web Depolama Alanı Sağlayıcısı Bağlayıcıları ve diğer DataFiles bağlantıları, joker karakter (***** ve **?**) kullanan filtre maskelerini desteklemez.*

Example 1: Bir dosya listesini yükleme

```
// 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: Diskte dosyaların listesini oluşturma

Bu örnek, Qlik Sense ile ilgili dosyaların tümünü bir klasöre yükler.

```
 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: Bir alanın değerleri aracılığıyla yineleme

Bu örnek, yüklenen FIELD değerlerinin listesi aracılığıyla yineleme yapar ve yeni bir alan (NEWFIELD) oluşturur. Her bir FIELD değeri için iki NEWFIELD kaydı oluşturulur.

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

Elde edilen tablo şöyle görünür:

Example table

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

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

if..then kontrol ifadesi, bir veya daha fazla mantıksal koşula bağlı olarak farklı yolları takip etmesi için kod yürütmesini zorlayan komut seçim yapısıdır.

Kontrol ifadeleri genellikle kod yürütme akışını kontrol etmek için kullanılır. Grafik ifadesinde bunun yerine **if** koşullu işlevini kullanın.

Söz Dizimi:

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

if..then deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, bu deyimin dört olası cümlesinin her biri (**if..then**, **elseif..then**, **else** ve **end if**) satır sınırını geçmemelidir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	True veya False olarak değerlendirilebilecek mantıksal bir ifade.
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.

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

Next kod anahtar sözcüğü **For** döngülerini kapatmak için kullanılır.

Sub..end sub

sub..end sub kontrol ifadesi, bir **call** deyimiyle çağrılabilecek bir alt yordam tanımlar.

Söz Dizimi:

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

Bağımsız değişkenler alt rutine kopyalanır ve **call** deyiminde karşılık gelen asıl parametre değişken adıysa, alt rutinden çıktıktan sonra tekrar dışarı kopyalanır.

Bir alt rutinin **call** deyimi ile aktarılan asıl parametrelerden daha fazla biçimsel parametresi varsa, ekstra parametreler NULL olarak başlatılır ve alt rutin içerisinde yerel değişken olarak kullanılabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
name	Alt rutinin adı.
paramlist	Alt rutinin biçimsel parametreleri için değişken adlarının virgülle ayrılmış listesi. Bunlar alt rutin içinde herhangi bir değişken gibi kullanılabilir.
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.

Sınırlamalar:

- **sub** deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, iki olası cümlesinin herhangi biri (**sub** ve **end sub**) satır sınırını geçmemelidir.
- Bir kontrol ifadesinde `Sub..end sub` ile bir alt rutin tanımladığınızda, ör. `if..then`, alt rutini yalnızca aynı kontrol ifadesinden çağırabilirsiniz.

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: - parametre aktarımı

```
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)
```

Yukarıdakilerin sonucunda yerel olarak, alt rutinin içinde, A 1 olarak başlatılır, B 4 olarak başlatılır ve C de NULL olarak başlatılır.

Alt rutinden çıkarken, A genel değişkeni değer olarak 2'yi alır (alt rutinden geri kopyalanır). İkinci gerçek parametre olan “(X+1)*2” bir değişken olmadığından, geri kopyalanmayacaktır. Son olarak, genel değişken C bu alt rutin çağrısından etkilenmez.

Switch..case..default..end switch

switch kontrol ifadesi, ifade değerine bağlı olarak, yolları takip etmek için kod yürütmesini zorlayan bir kod seçim yapısıdır.

Söz Dizimi:

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

> ***switch*** *deyimi bir kontrol ifadesi olduğundan ve bu nedenle noktalı virgülle veya satır sonuyla bittiğinden, bu deyimin dört olası cümlesinin her biri (**switch**, **case**, **default** ve **end switch**) satır sınırını geçmemelidir.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expression	Rastgele seçilen bir ifade.
valuelist	İfade değerinin karşılaştırılacağı virgülle ayrılmış değerler listesi. Kodun yürütülmesi, valuelist içindeki değeri expression içindeki değere eşit olup karşılaşılan ilk grupta yer alan deyimlerle devam eder. valuelist içindeki her değer rastgele bir ifade olabilir. Herhangi bir **case** cümlesinde eşleşme bulunmazsa, **default** cümlesi altındaki deyimler yürütülür (belirtilmişse).
statements	Bir veya daha fazla Qlik Sense kod deyiminden oluşan herhangi bir grup.

Örnek:

```
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

To kod anahtar sözcüğü çeşitli kod deyimlerinde kullanılır.

2.4 Kod önekleri

Önekler uygulanabilir durumdaki normal deyimlere uygulanabilir; ancak kontrol ifadelerine asla uygulanamaz. Bununla birlikte **when** ve **unless** önekleri birkaç belirli kontrol ifadesi cümlesinde sonek olarak kullanılabilir.

Tüm kod anahtar sözcükleri küçük harf ve büyük harften oluşan karakterlerin herhangi bir bileşimiyle yazılabilir. Bununla birlikte, deyimlerde kullanılan alan ve değişken adları büyük/küçük harf duyarlıdır.

Kod öneklerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Add

Add öneki, başka bir tabloya kayıt eklemesi gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir. **Add** öneki bir **Map** deyiminde de kullanılabilir.

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

Buffer

QVD dosyaları **buffer** önekiyle otomatik olarak oluşturulabilir ve korunabilir. Bu önek, koddaki çoğu **LOAD** ve **SELECT** deyiminde kullanılabilir. QVD dosyalarının deyimin sonucunu önbelleğe/arabelleğe almak için kullanıldığını belirtir.

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

Concatenate

Birleştirilmesi gerek iki tablo farklı alan kümelerine sahipse, bu iki tablonun birleştirilmesi yine de **Concatenate** önekiyle zorlanabilir.

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

Crosstable

crosstable öneki, bir çapraz tabloyu düz tabloya dönüştürmek için kullanılır. Yani, birçok sütun içeren geniş bir tablo, sütun başlıklarının tek bir öznitelik sütununa yerleştirildiği uzun bir tabloya dönüştürülür.

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

First

Bir **First** veya **LOAD** deyimine yönelik **SELECT (SQL)** öneki, bir veri kaynağı tablosundan maksimum sayıda kayıt kümesi yüklemek için kullanılır.

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

Generic

Genel veritabanlarının açılması ve yüklenmesi **generic** önekiyle yapılabilir.

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

Hierarchy

hierarchy öneki, üst-alt öğe hiyerarşi tablosunu Qlik Sense veri modelinde faydalı bir tabloya dönüştürmek için kullanılır. Bu önek, **LOAD** veya **SELECT** deyiminin önüne konulabilir ve yüklenen deyimin sonucunu tablo dönüştürme için girdi olarak kullanır.

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

HierarchBelongsTo

Bu önek, üst-alt öğe hiyerarşi tablosunu Qlik Sense veri modelinde faydalı bir tabloya dönüştürmek için kullanılır. Bu önek, **LOAD** veya **SELECT** deyiminin önüne konulabilir ve yüklenen deyimin sonucunu tablo dönüştürme için girdi olarak kullanır.

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

Inner

join ve **keep** öneklerinin öncesinde **inner** öneki gelebilir. Bu önek, **join** önekinden önce kullanılırsa, bir iç birleştirme kullanılması gerektiğini belirtir. Sonuç olarak ortaya çıkan tablo, bu nedenle, yalnızca bağlantılı alan değerlerinin her iki tabloda da temsil edildiği ham veri tablolarından alan değer kombinasyonlarını içerir. Bu önek, **keep** önekinden önce kullanılırsa, Qlik Sense içinde saklanmadan önce her iki ham veri tablosunun ortak kesiştiği noktaya azaltılması gerektiğini belirtir. .

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

IntervalMatch

Genişletilmiş **IntervalMatch** öneki, ayrık sayısal değerleri bir veya daha fazla sayısal aralıkla eleştiren ve isteğe bağlı olarak bir veya daha fazla ek anahtarın değerlerini eşleştiren bir tablo oluşturmak için kullanılır.

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

Join

join öneki, yüklenmiş tabloyu mevcut adlandırılmış bir tabloyla veya daha önce oluşturulmuş son veri tablosuyla birleştirir.

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

Keep

keep öneki, **join** önekine benzerdir. Aynı **join** öneki gibi, yüklenen tabloyu var olan bir adlandırılmış tablo veya daha önce oluşturulan son veri tablosu ile karşılaştırır, ancak yüklenen tabloyu var olan bir tablo ile birleştirmek yerine, Qlik Sense içinde depolanmadan önce, tablo verilerinin kesişimine bağlı olarak iki

tablonun birini ya da her ikisini birden indirgeme etkisine sahiptir. Karşılaştırma işlemi, ortak alanların üzerinden yapılan doğal birleştirmeye eşdeğerdir; yani, karşılık gelen birleştirme işlemiyle aynıdır. Ancak, iki tablo birleştirilmez ve Qlik Sense içinde iki ayrı ayrı adlandırılmış tablo olarak saklanır.

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

Left

Join ve **Keep** öneklerinin öncesinde **left** öneki gelebilir.

Bu önek, **join** önekinden önce kullanılırsa, sol birleştirme kullanılması gerektiğini belirtir. Sonuç olarak ortaya çıkan tablo yalnızca, bağlı alan değerlerinin ilk tabloda temsil edildiği ham veri tablolarından alan değerleri birleşimlerini içerir. **keep** öğesinden önce kullanılması durumunda, ikinci ham veri tablonun Qlik Sense içinde depolanmadan önce birinci tabloyla ortak kesişimine azaltılması gerektiğini belirtir.

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

Mapping

mapping öneki, örneğin kod yürütme sırasında alan değerlerini ve alan adlarını değiştirmek için kullanılabilecek bir eşleme tablosu oluşturmak için kullanılır.

```
Eşleme  ( loadstatement | selectstatement )
```

Merge

Merge öneki, yüklenen tablonun başka bir tabloyla birleştirilmesi gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir.

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

NoConcatenate

NoConcatenate öneki, aksi takdirde otomatik olarak birleştirilecek olan, birbiriyle alan kümelere sahip yüklenmiş iki tablonun iki ayrı dahili tablo olarak işlenmesini zorlar.

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

Outer

Açık **Join** öneki, bir dış birleştirmeyi belirtmek için **Outer** önekinden önce gelebilir. Bir dış birleştirmede iki tablo arasındaki tüm bileşimler oluşturulur. Bu nedenle, sonuç olarak ortaya çıkan tablo, bağlantılı alan değerlerinin bir tabloda veya her iki tabloda da temsil edildiği ham veri tablolarından alan değer birleşimlerini içerir. **Outer** anahtar sözcüğü isteğe bağlıdır ve bir birleştirme öneki belirtilmediğinde kullanılan varsayılan birleştirme türüdür.

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

Partial reload

Bir tam yeniden yükleme mevcut veri modelindeki tüm tabloları silerek başlar, ardından yükleme kodunu çalıştırır. *Kısmi yeniden yükleme (page 60)* bunu yapmaz. Bunun yerine, tüm tabloları veri modelinde tutar ve ardından yalnızca bir **Add**, **Merge** veya **Replace** öneki olan **Load** ve **Select** deyimlerini yürütür. Diğer

veri tabloları komuttan etkilenmez. **only** bağımsız değişkeni, deyimin yalnızca kısmi yeniden yüklemeler sırasında yürütülmesi, tam yüklemeler sırasında yoksayılması gerektiğini belirtir. Aşağıdaki tablo, kısmi ve tam yeniden yüklemeler için deyim yürütmeyi özetler.

Replace

Replace öneki, yüklenen tablonun başka bir tablonun yerini alması gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir. **Replace** öneki bir **Map** deyiminde de kullanılabilir.

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

Right

Join ve **Keep** öneklerinin öncesinde **right** öneki gelebilir.

Bu önek, **join** önekinden önce kullanılırsa, sağ birleştirme kullanılması gerektiğini belirtir. Sonuç olarak elde edilen tablo yalnızca, bağlayıcı alan değerlerinin ikinci tabloda temsil edildiği ham veri tablolarına ait alan değerlerinin bileşimlerini içerir. **keep** öğesinden önce kullanılması durumunda, birinci ham veri tablosunun Qlik Sense içinde depolanmadan önce ikinci tabloyla ortak kesişimine azaltılması gerektiğini belirtir.

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

Sample

Bir **LOAD** veya **SELECT** deyimine yönelik **sample** öneki, veri kaynağından rastgele sayıda kayıt yüklemek için kullanılır.

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

Semantic

Kayıtlar arasında ilişki içeren tablolar bir **semantic** önekiyle yüklenebilir. Bu örneğin, bir kaydın bir diğerine işaret ettiği (üst öğe, aittir veya öncel gibi), bir tablo içindeki kendi kendine başvurular olabilir.

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

Unless

unless öneki ve soneki bir deyimin veya bir çıkış cümlesinin değerlendirilip değerlendirilmemesi gerektiğini belirleyen koşullu bir cümle oluşturmak için kullanılır. Bu, uzun **if..end if** deyiminin kısa bir alternatifi olarak da görülebilir.

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

When

when öneki ve soneki bir deyimin veya bir çıkış cümlesinin yürütülüp yürütülmemesi gerektiğini belirleyen koşullu bir cümle oluşturmak için kullanılır. Bu, uzun **if..end if** deyiminin kısa bir alternatifi olarak da görülebilir.

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

Add

Add öneki, başka bir tabloya kayıt eklemesi gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir. **Add** öneki bir **Map** deyiminde de kullanılabilir.

> *Kısmi yeniden yüklemenin düzgün çalışması için, kısmi yeniden yükleme tetiklenmeden önce uygulamanın verilerle açılması gerekir.*

Yeniden Yükle düğmesini kullanarak kısmi yeniden yükleme gerçekleştirin. Qlik Engine JSON API öğesini de kullanabilirsiniz.

Söz Dizimi:

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

```
Add [only] mapstatement
```

Normal (kısmi olmayan) bir yeniden yükleme sırasında, **Add LOAD** yapısı normal bir **LOAD** ifadesi olarak çalışacaktır. Kayıtlar oluşturulur ve bir tabloda saklanır.

Concatenate öneki kullanılıyorsa veya aynı alan kümesine sahip bir tablo varsa, kayıtlar ilgili mevcut tabloya eklenir. Aksi takdirde **Add LOAD** yapısı yeni bir tablo oluşturur.

Kısmi yeniden yükleme aynı şeyi yapar. Tek fark, **Add LOAD** yapısının asla yeni bir tablo oluşturmamasıdır. Her zaman, önceki kod yürütme işleminden, kayıtların eklenmesi gereken ilgili bir tablo vardır.

Çoğaltma için denetim gerçekleştirilmez. Bu yüzden, **Add** öneki kullanan bir deyim çoğu zaman çoğaltmaları koruyan bir distinct niteleyicisi veya bir where cümlesi içerir.

Add Map...Using deyimi, eşlemenin kısmi kod yürütmesi sırasında da gerçekleştirilmesine neden olur.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
only	Deyimin sadece kısmi yeniden yüklemeler sırasında yürütülmesi gerektiğini belirten isteğe bağlı bir niteleyici. Normal (kısmi olmayan) yeniden yüklemeler sırasında dikkate alınmamalıdır.

Örnekler ve sonuçlar:

Örnek	Sonuç
Tab1: LOAD Name, Number FROM Persons.csv; Add LOAD Name, Number FROM newPersons.csv;	Normal yeniden yükleme sırasında, veriler *Persons.csv* dosyasından yüklenir ve Tab1 Qlik Sense tablosunda depolanır. *NewPersons.csv* dosyasından alınan veriler, ardından aynı Qlik Sense tablosuna birleştirilir. Kısmi yeniden yükleme sırasında, veriler *NewPersons.csv* dosyasından yüklenir ve Tab1 Qlik Sense tablosunun sonuna eklenir. Çoğaltma denetimi gerçekleştirilmez.
Tab1: SQL SELECT Name, Number FROM Persons.csv; Add LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Çoğaltmalar denetimi, Name öğesinin daha önceden yüklenmiş tablo verilerinde var olup olmadığına bakılarak gerçekleştirilir. Normal yeniden yükleme sırasında, veriler *Persons.csv* dosyasından yüklenir ve Tab1 Qlik Sense tablosunda depolanır. *NewPersons.csv* dosyasından alınan veriler, ardından aynı Qlik Sense tablosuna birleştirilir. Kısmi yeniden yükleme sırasında, veriler *NewPersons.csv* Qlik Sense tablosunun sonuna eklenen Tab1 dosyasından yüklenir. Çoğaltmalar denetimi, Name öğesinin daha önceden yüklenmiş tablo verilerinde var olup olmadığına bakılarak gerçekleştirilir.
Tab1: LOAD Name, Number FROM Persons.csv; Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Normal yeniden yükleme sırasında, veriler *Persons.csv* dosyasından yüklenir ve Tab1 Qlik Sense tablosunda depolanır. *NewPersons.csv* dosyasını yükleyen deyim göz ardı edilir. Kısmi yeniden yükleme sırasında, veriler *NewPersons.csv* Qlik Sense tablosunun sonuna eklenen Tab1 dosyasından yüklenir. Çoğaltmalar denetimi, Name öğesinin daha önceden yüklenmiş tablo verilerinde var olup olmadığına bakılarak gerçekleştirilir.

Buffer

QVD dosyaları **buffer** önekiyle otomatik olarak oluşturulabilir ve korunabilir. Bu önek, koddaki çoğu **LOAD** ve **SELECT** deyiminde kullanılabilir. QVD dosyalarının deyimin sonucunu önbelleğe/arabelleğe almak için kullanıldığını belirtir.

Söz Dizimi:

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

Bir seçenek kullanılmazsa, kodun ilk yürütülmesiyle oluşturulan QVD belleği süresiz olarak kullanılır.

Arabellek dosyası, genellikle *C:\ProgramData\Qlik\Sense\Engine\Buffers* (sunucu yüklemesi) veya *C:\Kullanıcılar\{user}\Belgeler\Qlik\Sense\Buffers* (Qlik Sense Desktop) olan *Arabellekler* alt klasöründe depolanır.

QVD dosyasının adı hesaplanan bir addır, yani takip eden **LOAD** veya **SELECT** deyiminin tamamının veya diğer ayırıcı bilgilerin 160 bit onaltılık karmasıdır. Bu, QVD belleğinin, takip eden **LOAD** veya **SELECT** deyimindeki herhangi bir değişiklikle geçersiz kılınacağı anlamına gelir.

QVD bellekleri normalde, oluşturduğu uygulamadaki tam kod yürütme boyunca herhangi bir konumda artık kendisine referansta bulunulmadığında veya oluşturduğu uygulama artık var olmadığında kaldırılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
artışlı	incremental seçeneği, temel bir dosyanın yalnızca bir bölümünü okuma özelliğini etkinleştirir. Dosyanın önceki boyutu, QVD dosyasının XML üst bilgisinde depolanır. Bu, özellikle günlük dosyalarıyla kullanışlıdır. Önceki bir durumda yüklenen tüm kayıtlar QVD dosyasından okunurken, takip eden yeni kayıtlar orijinal kaynaktan okunur ve son olarak güncelleştirilmiş bir QVD dosyası oluşturulur. incremental seçeneği yalnızca **LOAD** cümleleri ve metin dosyalarıyla kullanılabilir. Eski verilerin değiştirildiği veya silindiği durumlarda artımlı yükleme kullanılamaz.
stale [after] amount [(days | hours)]	amount, zaman dönemini belirten bir sayıdır. Ondalıklar kullanılabilir. Atlandığında birimin günler olduğu varsayılır. stale after seçeneği, orijinal verilerde basit bir zaman damgasının bulunmadığı durumlarda tipik olarak veritabanı kaynaklarıyla kullanılır. Bunun yerine, kullanılacak QVD anlık görüntüsünün ne kadar eski olabileceğini belirtirsiniz. Stale after cümlesi, basit bir şekilde, QVD belleğinin oluşturulma zamanından başlayan ve sonrasında geçerli sayılmayacağı bir zaman dönemi belirtir. Bu zamandan önce QVD belleği veriler için kaynak olarak kullanılır ve bundan sonra orijinal veri kaynağı kullanılır. Bu durumda QVD bellek dosyası otomatik olarak güncelleştirilir ve yeni bir dönem başlar.

Sınırlamalar:

Çeşitli sınırlamalar mevcuttur; bunlardan en önemlisi, herhangi bir karmaşık deyimin çekirdeğinde bir dosya **LOAD** veya **SELECT** deyimi olması gerekliliğidir.

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

Birleştirilmesi gerek iki tablo farklı alan kümelerine sahipse, bu iki tablonun birleştirilmesi yine de **Concatenate** önekiyle zorlanabilir. Bu deyim, var olan bir adlandırılmış tabloyla ya da daha önce oluşturulmuş son mantıksal tabloyla birleştirmeyi zorlar.

Söz Dizimi:

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

Birleştirme deyimi prensipte **SQL UNION** deyimiyle aynıdır, ancak iki fark vardır:

- **Concatenate** öneki, tabloların birebir aynı alan adlarına sahip olup olmamasına bakılmaksızın kullanılabilir.
- **Concatenate** öneki ile birebir aynı olan kayıtlar çıkarılmaz.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Var olan tablonun adı.

Örnek:

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

Crosstable

crosstable öneki, bir çapraz tabloyu düz tabloya dönüştürmek için kullanılır. Yani, birçok sütun içeren geniş bir tablo, sütun başlıklarının tek bir öznitelik sütununa yerleştirildiği uzun bir tabloya dönüştürülür.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
attribute field name	Öznitelik değerlerini içeren alan.
data field name	Veri değerlerini içeren alan.
n	Genel şekle dönüştürülecek tablonun öncesinde gelen niteleyici alanlarının sayısı. Varsayılan 1'dir.

Çapraz tablo, biri sütun üst bilgileri olarak kullanılan, üst bilgi verilerinden oluşan iki veya daha fazla dikey liste arasında bir değerler matrisi içeren yaygın bir tablo türüdür. Bunun tipik bir örneği her ay için bir sütun bulundurmak olabilir. **crosstable** önekinin sonucu şu olur: Sütun üst bilgileri (örneğin ay adları) bir alanda (öznitelik alanında) ve sütun verileri (ay sayıları) ikinci bir alanda (veri alanında) saklanır.

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Sonuç

Sonuç tablosu

Product	Oca 2021	Şub 2021	Mar 2021	Nis 2021	May 2021	Haz 2021
A	100	98	103	63	108	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Açıklama

Bu örnek, her ay için bir sütun ve her ürün için bir satır içeren bir çapraz tabloyu göstermektedir. Mevcut biçiminde, bu verilerin analiz edilmesi kolay değildir. Tüm rakamların bir alanda ve tüm ayların başka bir alanda, yani üç sütunlu bir tabloda olması çok daha iyi olurdu. Şimdi, çapraz tabloya böyle bir dönüşümün nasıl yapıldığını görelim.

Çapraz tabloyu dönüştürme

Koddaki yorumu kaldırın ve çalıştırın.

```
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;
```

Sonuç

Sonuç tablosu

Product	Ay	Satışlar
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

Açıklama

Çapraz tablo, Ay için bir sütun ve Satış için bir başka sütun içeren düz tabloya dönüştürülür.

First

Bir **First** veya **LOAD** deyimine yönelik **SELECT (SQL)** öneki, bir veri kaynağı tablosundan maksimum sayıda kayıt kümesi yüklemek için kullanılır.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n	Okunacak maksimum kayıt sayısını gösteren bir tamsayı olarak değerlendiren rastgele seçilmiş ifade. *n* parantez içine alınabilir (*(n)* gibi), ancak bu gerekli değildir.

Örnekler:

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

Generic

Genel veritabanlarının açılması ve yüklenmesi **generic** önekiyle yapılabilir.

Genel veritabanları/veri kaynakları, yapılandırılmış tekrarlayan veriler (örneğin, varlıkların benzer niteliklerle tekrar tekrar tanımlandığı bir adres listesi veya bir ürün özellikleri sayfası gibi) içerir.

Söz Dizimi:

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

Örnekler:

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

generic deyimiyle yüklenen tablolar otomatik olarak birleştirilmez.

Örnek

1. Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

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

US, Name, AAA
US, Address, A123
```

```
US, Phone, 001-123

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

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

Sonuç

Sonuç tablosu

Bölge	Ad	Address	Telefon
SWE	CCC	C7789	003-789
US	AAA	A123	001-123
US	AAA	A123	002-456
US	AAA	B456	001-123
US	AAA	B456	002-456
US	BBB	A123	001-123
US	BBB	A123	002-456
US	BBB	B456	001-123
US	BBB	B456	002-456

2. Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
Sheet1:
Generic Load * inline [
object, attribute, value
ball, color, red
ball, diameter, 10 cm
ball, weight, 100 g
box, color, black
box, height, 16 cm
box, length, 20 cm
box, weight, 500 g
box, width, 10 cm ];
```

Sonuç

Sonuç tablosu

nesne	renk	çap	uzunluk	yükseklik	genişlik	ağırlık
ball	red	10 cm	-	-	-	100 g
box	black	-	20 cm	16 cm	10 cm	500 g

Hierarchy

hierarchy öneki, üst-alt öğe hiyerarşi tablosunu Qlik Sense veri modelinde faydalı bir tabloya dönüştürmek için kullanılır. Bu önek, **LOAD** veya **SELECT** deyiminin önüne konulabilir ve yüklenen deyimin sonucunu tablo dönüştürme için girdi olarak kullanır.

Önek genişletilmiş bir düğüm tablosu oluşturur; bu tablo normalde giriş tablosuyla aynı sayıda kayda sahiptir, ancak buna ek olarak hiyerarşideki her seviye ayrı bir alanda saklanır. Yol alanı bir ağaç yapısında kullanılabilir.

Söz Dizimi:

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

Giriş tablosu bir bitişik düğüm tablosu olmalıdır. Bitişik düğüm tabloları, her bir kaydın bir düğüme karşılık geldiği ve ana düğüme bir referans içeren bir alana sahip olduğu tablolardır. Böyle bir tabloda düğüm yalnızca bir kayıtta saklanır, ancak düğüm birden fazla alt öğeye sahip olmaya devam edebilir. Tablo, doğal olarak, düğümlerin özniteliklerini tanımlayan ek alanlar içerebilir.

Önek genişletilmiş bir düğüm tablosu oluşturur; bu tablo normalde giriş tablosuyla aynı sayıda kayda sahiptir, ancak buna ek olarak hiyerarşideki her seviye ayrı bir alanda saklanır. Yol alanı bir ağaç yapısında kullanılabilir.

Genellikle, giriş tablosu her bir düğüm için tam olarak bir kayda sahiptir ve böyle bir durumda çıkış tablosu aynı sayıda kaydı içerir. Bununla birlikte, bazen kimi zaman birden fazla ana öğeye sahip düğümler olabilir; yani bir düğüm giriş tablosunda birden fazla kayıtla temsil edilir. Bu durum söz konusuysa, çıkış tablosu giriş tablosundan daha fazla kayda sahip olabilir.

Düğüm kimliği sütununda ana kimliği bulunmayan tüm düğümler (ana kimliği eksik düğümler de dahil) kök olarak kabul edilir. Ayrıca, yalnızca kök düğümle bağlantısı (doğrudan ya da dolaylı) olan düğümler yüklenir ve böylece döngüsel referansların önüne geçilir.

Ana düğüm adını, düğümün yolunu ve düğüm derinliğini içeren ek alanlar oluşturulabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
NodeID	Düğüm kimliğini içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
ParentID	Ana düğümün düğüm kimliğini içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
NodeName	Düğümün adını içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
ParentName	Yeni **ParentName** alanını adlandırmak için kullanılan dize. Atlandığı takdirde bu alan oluşturulmaz.
ParentSource	Düğüm yolunu oluşturmak için kullanılan düğümün adını içeren alanın adı. İsteğe bağlı parametre. Atlandığı takdirde **NodeName** kullanılır.
PathName	Kökten düğüme giden yolu içeren yeni **Path** alanını adlandırmak için kullanılan dize. İsteğe bağlı parametre. Atlandığı takdirde bu alan oluşturulmaz.
PathDelimiter	Yeni **Path** alanında sınırlayıcı olarak kullanılan dize. İsteğe bağlı parametre. Atlandığı takdirde, '/' kullanılır.
Depth	Hiyerarşideki düğümün derinliğini içeren yeni **Depth** alanını adlandırmak için kullanılan dize. İsteğe bağlı parametre. Atlandığı takdirde bu alan oluşturulmaz.

Örnek:

```
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

Bu önek, üst-alt öğe hiyerarşi tablosunu Qlik Sense veri modelinde faydalı bir tabloya dönüştürmek için kullanılır. Bu önek, **LOAD** veya **SELECT** deyiminin önüne konulabilir ve yüklenen deyimin sonucunu tablo dönüştürme için girdi olarak kullanır.

Bu önek hiyerarşinin tüm üst-alt ilişkilerini içeren bir tablo oluşturur. Böylece üst öğe alanları, hiyerarşideki bütün ağaçları seçmek için kullanılabilir. Çıkış tablosu çoğu durumda her düğüm için çok sayıda kayıt içerebilir.

Söz Dizimi:

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

Giriş tablosu bir bitişik düğüm tablosu olmalıdır. Bitişik düğüm tabloları, her bir kaydın bir düğüme karşılık geldiği ve ana düğüme bir referans içeren bir alana sahip olduğu tablolardır. Böyle bir tabloda düğüm yalnızca bir kayıtta saklanır, ancak düğüm birden fazla alt öğeye sahip olmaya devam edebilir. Tablo, doğal olarak, düğümlerin özniteliklerini tanımlayan ek alanlar içerebilir.

Bu önek hiyerarşinin tüm üst-alt ilişkilerini içeren bir tablo oluşturur. Böylece üst öğe alanları, hiyerarşideki bütün ağaçları seçmek için kullanılabilir. Çıkış tablosu çoğu durumda her düğüm için çok sayıda kayıt içerebilir.

Düğümlerin derinlik farklılıklarını içeren ek bir alan oluşturulabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
NodeID	Düğüm kimliğini içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
ParentID	Ana düğümün düğüm kimliğini içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
NodeName	Düğümün adını içeren alanın adı. Bu alan giriş tablosunda bulunmalıdır.
AncestorID	Üst düğüm kimliğini içeren yeni üst öğe kimliği alanını adlandırmak için kullanılan dize.
AncestorName	Üst düğümün adını içeren yeni üst öğe alanını adlandırmak için kullanılan dize.
DepthDiff	Üst düğüme göre hiyerarşideki düğümün derinliğini içeren yeni **DepthDiff** alanını adlandırmak için kullanılan dize. İsteğe bağlı parametre. Atlandığı takdirde bu alan oluşturulmaz.

Örnek:

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

Results

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

Inner

join ve **keep** öneklerinin öncesinde **inner** öneki gelebilir. Bu önek, **join** önekinden önce kullanılırsa, bir iç birleştirme kullanılması gerektiğini belirtir. Sonuç olarak ortaya çıkan tablo, bu nedenle, yalnızca bağlantılı alan değerlerinin her iki tabloda da temsil edildiği ham veri tablolarından alan değer kombinasyonlarını içerir. Bu önek, **keep** önekinden önce kullanılırsa, Qlik Sense içinde saklanmadan önce her iki ham veri tablosunun ortak kesiştiği noktaya azaltılması gerektiğini belirtir.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç

Sonuç tablosu

Column1	Column2	Column3
A	B	C
1	aa	xx

Açıklama

Bu örnek, yalnızca hem birinci (sol) hem de ikinci (sağ) tablolarda bulunan değerlerin birleştirildiği Inner Join çıktısını gösterir.

IntervalMatch

Genişletilmiş **IntervalMatch** öneki, ayrık sayısal değerleri bir veya daha fazla sayısal aralıkla eleştiren ve isteğe bağlı olarak bir veya daha fazla ek anahtarın değerlerini eşleştiren bir tablo oluşturmak için kullanılır.

Söz Dizimi:

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

IntervalMatch öneki, aralıkları yükleyen bir **LOAD** veya **SELECT** deyiminden önce yerleştirilmelidir. Ayrık veri noktalarını içeren alan (aşağıdaki örnekte Zaman) ve ek anahtarlar, **IntervalMatch** önekinin bulunduğu deyimden önce Qlik Sense içine zaten yüklenmiş olmalıdır. Önek veritabanı tablosundan bu alanı tek başına okumaz. Önek, yüklenmiş aralıklar ve anahtarlar tablosunu ek sütun (ayrık sayısal veri noktaları) içeren tabloya dönüştürür. Bu işlem, aynı zamanda yeni tablo ayrık veri noktasının, aralığın ve anahtar alanlarının değerinin her olası kombinasyonu için bir kayıt içerecek şekilde kayıt sayısını genişletir

Aralıklar çakışabilir ve ayrık değerler tüm eşleşen aralıklara bağlanır.

IntervalMatch öneki anahtar alanlarıyla genişletildiğinde, ayrık sayısal değerleri bir veya daha fazla sayısal aralıkla eşleştirirken aynı zamanda bir veya daha fazla ek anahtarın değerlerini eşleştiren tablo oluşturmak için kullanılır.

Tanımlanmamış aralık sınırlarının göz ardı edilmesini önlemek için, NULL değerlerin aralığın alt ve üst sınırlarını oluşturan diğer alanlara eşlenmesine izin vermek gerekebilir. Bu da, NULL değerleri ayrık sayısal veri noktalarının herhangi birinden çok önce veya sonra olacak şekilde sayısal bir değerle değiştiren açık bir test ya da **NullAsValue** deyimi ile başarılabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
matchfield	Aralıklara bağlanacak ayrık sayısal değerleri içeren alan.
keyfield	Dönüştürme sırasında eşleştirilecek ek öznitelikleri içeren alan.
loadstatement orselectstatement	Sonuçta birinci alanı her bir aralığın alt sınırını içeren, ikinci alanı her bir aralığın üst sınırını içeren ve anahtar eşleştirmesi kullanılması durumunda da üçüncü ve varsa sonraki alanları **IntervalMatch** deyiminde bulunan anahtar alanları içeren bir tablo ortaya çıkmalıdır. Aralıklar her zaman kapalıdır; yani uç noktaları her zaman aralığa dahil edilir. Sayısal olmayan sınırlar, aralığı göz ardı edilmiş (tanımlanmamış) olarak işler.

Example 1:

Aşağıdaki iki tabloda, ilki birkaç ayrık olayı listelerken, ikincisi farklı siparişlerin üretiminin başlangıç ve bitiş zamanlarını tanımlar. **IntervalMatch** öneki aracılığıyla, örneğin hangi siparişlerin kesintilerden etkilendiğini ve hangi siparişlerin hangi vardiyalarda üretildiğini öğrenmek amacıyla iki tabloyu mantıksal olarak birbirine bağlamak mümkündür.

```
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;
```

Artık, **OrderLog** tablosu ek bir sütun içermektedir: *Time*. Kayıtların sayısı da genişlemiştir.

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: (keyfield kullanarak)

Yukarıdaki örnekle aynı olup anahtar alan olarak *ProductionLine* eklenmiştir.

```
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;
```

Artık aşağıdaki gibi bir tablo kutusu oluşturulabilir:

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

join öneki, yüklenmiş tabloyu mevcut adlandırılmış bir tabloyla veya daha önce oluşturulmuş son veri tablosuyla birleştirir.

Söz Dizimi:

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

Birleştirme, tüm ortak alanlar üzerinde yapılan doğal bir birleştirmedir. Join deyiminden önce **inner**, **outer**, **left** veya **right** öneklerinden biri gelebilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek:

```
Join SELECT * from table1;



tab1:

LOAD * from file1.csv;

tab2:

LOAD * from file2.csv;

.. .. ..

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

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

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

Sonuç tablosu

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

Açıklama

Bu örnekte, iki tablo (Tablo1 ve Tablo2), Tablo1 etiketli tek bir tabloda birleştirilmiştir. Bu gibi durumlarda, **birleştirme** öneki genellikle tek bir tablonun değerleri üzerinde toplama gerçekleştirmek amacıyla birkaç tabloyu tek bir tabloda birleştirmek için kullanılır.

Keep

keep öneki, **join** önekine benzerdir. Aynı **join** öneki gibi, yüklenen tabloyu var olan bir adlandırılmış tablo veya daha önce oluşturulan son veri tablosu ile karşılaştırır, ancak yüklenen tabloyu var olan bir tablo ile birleştirmek yerine, Qlik Sense içinde depolanmadan önce, tablo verilerinin kesişimine bağlı olarak iki

tablonun birini ya da her ikisini birden indirgeme etkisine sahiptir. Karşılaştırma işlemi, ortak alanların üzerinden yapılan doğal birleştirmeye eşdeğerdir; yani, karşılık gelen birleştirme işlemiyle aynıdır. Ancak, iki tablo birleştirilmez ve Qlik Sense içinde iki ayrı ayrı adlandırılmış tablo olarak saklanır.

Söz Dizimi:

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

keep önekinin öncesinde **inner**, **left** veya **right** öneklerinden biri gelmelidir.

Qlik Sense kod dilinde açık **join** öneki, iki tablonun tam birleştirmesini gerçekleştirir. Sonuç bir tablodur. Birçok durumda, bu tür birleştirmeler çok büyük tabloların ortaya çıkmasıyla sonuçlanır. Qlik Sense uygulamasının ana özelliklerinden biri, birden fazla tabloyu birleştirmek yerine bu tablolar arasında ilişkilendirme yapma kabiliyetidir; bu da bellek kullanımını önemli oranda azaltır, işleme hızını artırır ve çok büyük bir esneklik sunar. Bu nedenle, Qlik Sense kodlarında açık birleştirmelerden genellikle kaçınılması gerekir. keep fonksiyonelliği, açık birleştirmelerin kullanılması gereken durumların sayısını azaltmak üzere tasarlanmıştır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek:

```
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

Join ve **Keep** öneklerinin öncesinde **left** öneki gelebilir.

Bu önek, **join** önekinden önce kullanılırsa, sol birleştirme kullanılması gerektiğini belirtir. Sonuç olarak ortaya çıkan tablo yalnızca, bağlı alan değerlerinin ilk tabloda temsil edildiği ham veri tablolarından alan değerleri birleşimlerini içerir. **keep** öğesinden önce kullanılması durumunda, ikinci ham veri tablonun Qlik Sense içinde depolanmadan önce birinci tabloyla ortak kesişimine azaltılması gerektiğini belirtir.

Aynı adı taşıyan dize fonksiyonunu mu arıyordunuz? Bkz. Left (page 798)

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç

Sonuç tablosu

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

Açıklama

Bu örnek, yalnızca ilk (sol) tabloda bulunan değerlerin birleştirildiği Left Join çıktısını gösterir.

Eşleme

mapping öneki, örneğin kod yürütme sırasında alan değerlerini ve alan adlarını değiştirmek için kullanılabilecek bir eşleme tablosu oluşturmak için kullanılır.

Söz Dizimi:

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

mapping öneki bir **LOAD** veya **SELECT** deyiminin önüne koyulabilir ve yükleme deyiminin sonucunu bir eşleme tablosu olarak saklar. Eşleme, kod yürütme sırasında alan değerlerinin ikame edilmesinde (örneğin, BD, B.D. veya Amerika yerine ABD kullanılması gibi) etkili bir yol sağlar. Bir eşleme tablosu, ilki karşılaştırma değerlerini ve ikincisi de istenen eşleme değerlerini içeren iki sütundan oluşur. Eşleme tabloları bellekte geçici olarak saklanır ve kod yürütmesinden sonra otomatik olarak bırakılır.

Eşleme tablosunun içeriğine örneğin, **Map … Using** deyimi, **Rename Field** deyimi, **Applymap()** fonksiyonu veya **Mapsubstring()** fonksiyonu kullanılarak erişilebilir.

Örnek:

Bu örnekte, ikamet ettikleri ülkeyi temsil eden ülke koduyla birlikte satış elemanlarının yer aldığı bir listeyi yüklüyoruz. Ülke kodunun yerine ülke adını koymak için, ülke kodunu ülkeyle eşleyen bir tablo kullanıyoruz. Eşleme tablosunda yalnızca üç ülke tanımlanmakta ve diğer ülke kodları 'Rest of the world' ile eşlenmektedir.

```
// 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';
```

Elde edilen tablo şöyle görünür:

Mapping table

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

Birleştirme

Merge öneki, yüklenen tablonun başka bir tabloyla birleştirilmesi gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir.

Tipik kullanım durumu, bir değişiklik günlüğü yüklediğiniz ve bunu kullanarak inserts, updates ve deletes öğesini var olan bir tabloya uygulamak istediğiniz zamanlardır.

Kısmi yeniden yüklemenin düzgün çalışması için, kısmi yeniden yükleme tetiklenmeden önce uygulamanın verilerle açılması gerekir.

Yeniden Yükle düğmesini kullanarak kısmi yeniden yükleme gerçekleştirin. Qlik Engine JSON API öğesini de kullanabilirsiniz.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
only	Deyimin sadece kısmi yeniden yüklemeler sırasında yürütülmesi gerektiğini belirten isteğe bağlı bir niteleyici. Deyim, normal (kısmi olmayan) yeniden yüklemelerde göz ardı edilir.
SequenceNoField	Bir zaman damgası içeren alanın adı veya işlemlerin sırasını tanımlayan bir sıra numarası.
SequenceNoVar	Birleştirilmekte olan tablonun SequenceNoField öğesi için maksimum değerin atandığı değişkenin adı.
ListOfKeys	Birincil anahtarı belirten, virgülle ayrılmış alan adları listesi.
Operation	Yükleme deyiminin ilk alanı işlemi bir metin dizesi olarak içermelidir: "Insert", "Update" veya "Delete". "i", "u" ve "d" de ayrıca kabul edilir.

Genel işlevsellik

Normal (kısmi olmayan) bir yeniden yükleme sırasında, **Merge LOAD** yapısı normal bir **Load** deyimi olarak; ancak eski kayıtları ve silinmek üzere işaretlenmiş kayıtları kaldırma ek işlevselliği ile çalışır. **Load** deyiminin ilk alanı işlemle ilgili bilgileri içermelidir: Insert, Update veya Delete.

Yüklenen her kayıt için kayıt tanımlayıcısı önceden yüklenen kayıtlarla karşılaştırılır ve yalnızca en son kayıt (sıra numarasına göre) saklanır. En son kayıt Delete ile işaretlenmişse hiçbiri saklanmaz.

Hedef tablo

Hangi tablonun değiştirileceği alan kümesi tarafından belirlenir. Aynı alan kümesine sahip (işlem olan ilk alan dışında) bir tablo zaten mevcutsa, değiştirilecek olan bu tablodur. Alternatif olarak tabloyu belirtmek için bir **Concatenate** ön eki belirtilebilir. Hedef tablo belirlenmezse, **Merge LOAD** yapısının sonucu yeni bir tabloda saklanır.

Concatenate ön eki kullanılırsa, ortaya çıkan tabloda mevcut tablo ile Merge işleminin girdisinin bileşimine karşılık gelen bir alan kümesi olur. Bu nedenle hedef tabloda Merge işleminin girdisi olarak kullanılan değişiklik günlüğünden daha fazla alan olabilir.

Kısmi bir yeniden yükleme tam bir yeniden yükleme ile aynı şeyi yapar. Tek fark kısmi bir yeniden yüklemenin seyrek olarak yeni bir tablo oluşturmasıdır. **Only** yan tümcesi kullanılmadığı sürece önceki komut dosyası yürütmedeki alan kümesine sahip olan bir hedef tablo her zaman mevcuttur.

Sıra numarası

Yüklenen değişiklik günlüğü birikmiş bir günlükse; yani zaten yüklenmiş olan değişiklikleri içeriyorsa, SequenceNoVar parametresi, giriş verilerinin miktarını sınırlamak için bir **Where** cümlesinde kullanılabilir. **Merge LOAD** daha sonra yalnızca SequenceNoField alanının şundan büyük olduğu kayıtları yüklemek için yapılabilir: SequenceNoVar. İşlem tamamlandığında **Merge LOAD**, SequenceNoVar değişkenine maksimum değeri SequenceNoField alanında görülen yeni bir değer atar.

İşlemler

Merge LOAD işleminde hedef tablodan daha az sayıda alan olabilir. Eksik alanlar farklı işlemlerde farklı şekilde işlenir:

Insert: **Merge LOAD** içinde eksik buna karşın hedef tabloda mevcut olan alanlar hedef tabloda NULL değerini alır.

Delete: Eksik alanlar sonucu etkilemez. İlgili kayıtlar yine de silinir.

Update: **Merge LOAD** içinde listelenen alanlar hedef tabloda güncellenir. Eksik alanlar değiştirilmez. Bu, aşağıdaki iki deyimin aynı olmadığı anlamına gelir:

- 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 ...;

Birinci deyim listelenen kayıtları günceller ve F2 değerini NULL ile değiştirir. İkincisi F2 değerini değiştirmez; bunun yerine değerleri hedef tabloda bırakır.

Merge LOAD işlemi joker karakterleri olan kayıtlarda; örneğin tüm değerleri göstermek yıldız içeren bir Section Access tablosunda kullanılamaz.

Örnekler

Örnek 1: Belirtilen tabloyla basit birleştirme

Bu örnekte, Persons adlı bir satır içi tablo üç satırla yüklenmektedir. **Merge** ardından tabloyu şu şekilde değiştirir:

- *Mary, 4* satırını ekler.
- *Steven, 3* satırını siler.
- *Jake*'e *5* sayısını atar.

Merge yürütüldükten sonra *LastChangeDate* değişkeni *ChangeDate* sütunundaki maksimum değere ayarlanır.

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç

Merge Load işleminden sonra oluşan tablo şu şekilde görünür:

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

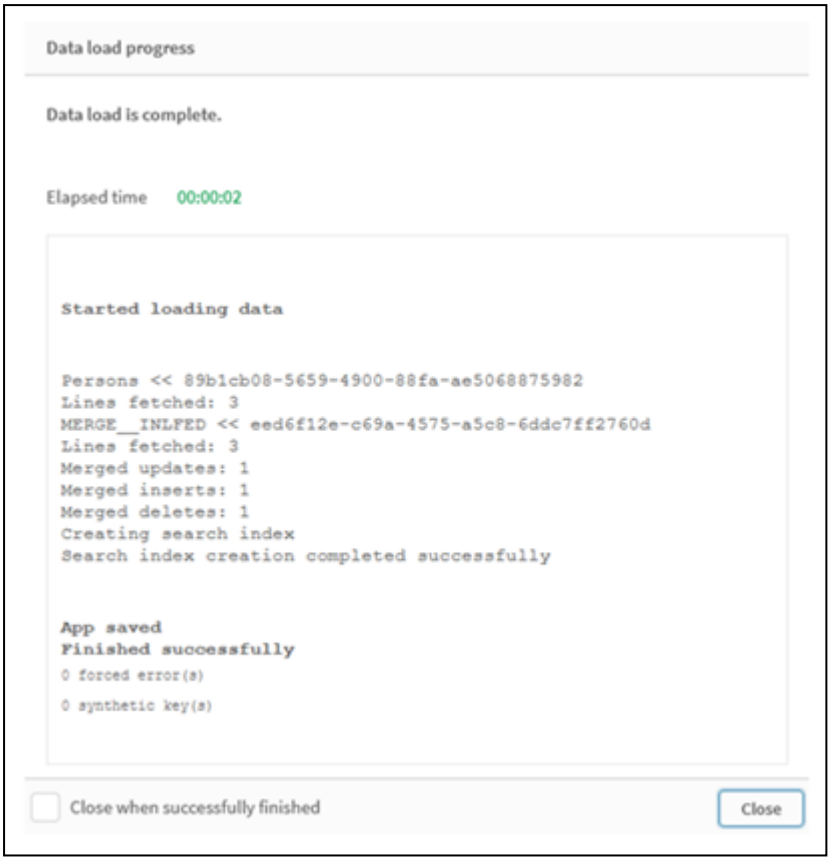
Merge Load işleminden sonra tablo şu şekilde görünür:

Resulting table

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

Veriler yüklendiğinde, yapılan işlemler **Veri yükleme ilerlemesi** diyalog penceresinde gösterilir.

Veri yükleme ilerlemesi diyalog penceresi



Örnek 2: Eksik alanları olan veri yükleme komut dosyası

Bu örnekte, yukarıdakiyle aynı veri yüklenmektedir, ancak bu kez her kişiye bir kimlik numarası verilerek.

Merge tabloyu aşağıdaki şekilde değiştirir:

- *Mary, 4* satırını ekler.
- *Steven, 3* satırını siler.
- *Jake*'e *5* sayısını atar.
- *Jill*'e *6* sayısını atar.

Komut dosyası

Burada, biri "Insert" ve "Delete", diğeri ise "Update" işlemi için olmak üzere iki **Merge Load** deyimi kullanıyoruz.

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç

Merge Load deyimlerinin ardından tablo şu şekilde görünür:

Resulting table

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

İkinci **Merge** deyiminin **Name** alanını içermediğine ve bunun sonucunda adların değiştirilmemiş olduğuna dikkat edin.

Örnek 3: Veri yükleme komut dosyası - ChangeDate ile bir Where yan tümcesi kullanan kısmi yeniden yükleme

Aşağıdaki örnekte **Only** bağımsız değişkeni **Merge** komutunun yalnızca kısmi bir yeniden yükleme sırasında yürütüldüğünü belirtir. Güncellemeler daha önce yakalanan LastChangeDate'e göre filtrelenir. **Merge** işlemi tamamlandıktan sonra LastChangeDate değişkenine birleştirme sırasında işlenen ChangeDate sütununun maksimum değeri atanır.

Komut dosyası

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

NoConcatenate

NoConcatenate öneki, aksi takdirde otomatik olarak birleştirilecek olan, birbiriyle alan kümelere sahip yüklenmiş iki tablonun iki ayrı dahili tablo olarak işlenmesini zorlar.

Söz Dizimi:

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

Örnek:

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

Only

Only kod anahtar sözcüğü bir toplama işlevi olarak veya **Add**, **Replace** ve **Merge** kısmi yeniden yükleme öneklerinde söz diziminin parçası olarak kullanılır.

Outer

Açık **Join** öneki, bir dış birleştirmeyi belirtmek için **Outer** önekinden önce gelebilir. Bir dış birleştirmede iki tablo arasındaki tüm bileşimler oluşturulur. Bu nedenle, sonuç olarak ortaya çıkan tablo, bağlantılı alan değerlerinin bir tabloda veya her iki tabloda da temsil edildiği ham veri tablolarından alan değer

birleşimlerini içerir. **Outer** anahtar sözcüğü isteğe bağlıdır ve bir birleştirme öneki belirtilmediğinde kullanılan varsayılan birleştirme türüdür.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç tablosu

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

Açıklama

Bu örnekte, iki tablo (Tablo1 ve Tablo2), Tablo1 etiketli tek bir tabloda birleştirilmiştir. Bu gibi durumlarda, **dış** önek genellikle tek bir tablonun değerleri üzerinde toplama gerçekleştirmek amacıyla birkaç tabloyu tek bir tablo olarak birleştirmek için kullanılır.

Kısmi yeniden yükleme

Bir tam yeniden yükleme mevcut veri modelindeki tüm tabloları silerek başlar, ardından yükleme kodunu çalıştırır.

Kısmi yeniden yükleme bunu yapmaz. Bunun yerine, tüm tabloları veri modelinde tutar ve ardından yalnızca bir **Add**, **Merge** veya **Replace** öneki olan **Load** ve **Select** deyimlerini yürütür. Diğer veri tabloları komuttan etkilenmez. **only** bağımsız değişkeni, deyimin yalnızca kısmi yeniden yüklemeler sırasında yürütülmesi, tam yüklemeler sırasında yoksayılması gerektiğini belirtir. Aşağıdaki tablo, kısmi ve tam yeniden yüklemeler için deyim yürütmeyi özetler.

Deyim	Tam yeniden yükleme	Kısmi yeniden yükleme
Load ...	Deyim çalışır	Deyim çalışmaz
Add/Replace/Merge Load ...	Deyim çalışır	Deyim çalışır
Add/Replace/Merge Only Load ...	Deyim çalışmaz	Deyim çalışır

Tam yeni yüklemelere kıyasla kısmi yeniden yüklemelerin birkaç avantajı vardır:

- Yalnızca son değişen verilerin yüklenmesi gerektiğinden daha hızlıdır. Büyük veri setlerinde bu fark önemlidir.
- Daha az veri yüklendiğinden daha az bellek tüketilir.
- Kaynak verilerde yapılan sorgulamalar daha hızlı çalışarak ağ sorunları riskini azalttığından daha güvenilirdir.

Kısmi yeniden yüklemenin düzgün çalışması için, kısmi yeniden yükleme tetiklenmeden önce uygulamanın verilerle açılması gerekir.

Yeniden Yükle düğmesini kullanarak kısmi yeniden yükleme gerçekleştirin. Qlik Engine JSON API öğesini de kullanabilirsiniz.

Örnek

1. Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve bir kısmi yeniden yükleme işlemi yapın. Sonucu görmek için sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

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

Sonuç

Resulting table

Num	Count(Num)
11	1
12	1
13	1

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

Açıklama

Deyim yalnızca bir kısmi yeniden yükleme sırasında yürütülür. "distinct" ön eki çıkarılırsa, **Num** alanının sayısı sonraki her kısmi yeniden yüklemeden sonra artar.

2. Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin. Bir tam yeniden yükleme işlemi yapın ve sonucu görüntüleyin. Sonra bir kısmi yeniden yükleme işlemi yapın ve sonucu görüntüleyin. Sonuçları görmek için sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

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

Sonuç

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4
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

Açıklama

İlk tablo bir tam yeniden yükleme sırasında yüklenir, ikinci tablo ise bir kısmi yeniden yükleme sırasında ilk tablonun yerini alır.

Replace

Replace kod anahtar sözcüğü, dize fonksiyonu veya kısmi yeniden yüklemede önek olarak kullanılır.

Replace

Replace öneki, yüklenen tablonun başka bir tablonun yerini alması gerektiğini belirtmek için koddaki bir **LOAD** veya **SELECT** deyimine eklenebilir. Bu ayrıca, bu deyimin bir kısmi yeniden yüklemede çalıştırılması gerektiğini belirtir. **Replace** öneki bir **Map** deyiminde de kullanılabilir.

> *Kısmi yeniden yüklemenin düzgün çalışması için, kısmi yeniden yükleme tetiklenmeden önce uygulamanın verilerle açılması gerekir.*

Yeniden Yükle düğmesini kullanarak kısmi yeniden yükleme gerçekleştirin. Qlik Engine JSON API öğesini de kullanabilirsiniz.

Söz Dizimi:

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

```
Replace [only] mapstatement
```

Normal (kısmi olmayan) bir yeniden yükleme sırasında, **Replace LOAD** yapısı normal bir **LOAD** ifadesi olarak çalışacak, ancak öncesinde bir **Drop Table** olacaktır. Önce eski tablo kaldırılır, ardından kayıtlar oluşturulur ve yeni bir tablo olarak saklanır.

Concatenate öneki kullanılıyorsa veya aynı alan kümesine sahip bir tablo varsa, bu bırakılacak ilgili tablo olacaktır. Aksi takdirde bırakılacak bir tablo yoktur ve **Replace LOAD** yapısı normal bir **LOAD** ile aynı olur.

Kısmi yeniden yükleme aynı şeyi yapar. Tek fark, her zaman bir önceki kod yürütme işleminden bırakılacak bir tablo olmasıdır. **Replace LOAD** yapısı her zaman önce eski tabloyu bırakır, sonra yeni bir tane oluşturur.

Replace Map...Using deyimi, eşlemenin kısmi kod yürütmesi sırasında da gerçekleştirilmesine neden olur.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
only	Deyimin sadece kısmi yeniden yüklemeler sırasında yürütülmesi gerektiğini belirten isteğe bağlı bir niteleyici. Normal (kısmi olmayan) yeniden yüklemeler sırasında dikkate alınmamalıdır.

Örnekler ve sonuçlar:

Örnek	Sonuç
`Tab1: Replace LOAD * from File1.csv;`	Hem normal hem de kısmi yeniden yükleme sırasında Qlik Sense Tab1 tablosu başlangıçta bırakılır. Daha sonra, File1.csv dosyasından yeni veriler yüklenir ve Tab1 içinde depolanır.
`Tab1: Replace only LOAD * from File1.csv;`	Normal yeniden yükleme sırasında bu deyim göz ardı edilir. Kısmi yeniden yükleme sırasında, önceden Tab1 olarak adlandırılmış herhangi bir Qlik Sense tablosu başlangıçta bırakılır. Daha sonra, File1.csv dosyasından yeni veriler yüklenir ve Tab1 içinde depolanır.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Normal yeniden yükleme sırasında, File1.csv dosyası ilk olarak Qlik Sense Tab1 tablosuna okunur, ancak daha sonra hemen bırakılır ve File2.csv dosyasından yüklenen yeni verilerle değiştirilir. File1.csv dosyasından alınan tüm veriler kaybolur. Kısmi yeniden yükleme sırasında Qlik Sense Tab1 tablosunun tamamı başlangıçta bırakılır. Daha sonra File2.csv dosyasından yüklenen yeni verilerle değiştirilir.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Normal yeniden yükleme sırasında, veriler File1.csv dosyasından yüklenir ve Tab1 Qlik Sense tablosunda depolanır. File2.csv göz ardı edilir. Kısmi yeniden yükleme sırasında Qlik Sense Tab1 tablosunun tamamı başlangıçta bırakılır. Daha sonra File2.csv dosyasından yüklenen yeni verilerle değiştirilir. File1.csv dosyasından alınan tüm veriler kaybolur.

Right

Join ve **Keep** öneklerinin öncesinde **right** öneki gelebilir.

Bu önek, **join** önekinden önce kullanılırsa, sağ birleştirme kullanılması gerektiğini belirtir. Sonuç olarak elde edilen tablo yalnızca, bağlayıcı alan değerlerinin ikinci tabloda temsil edildiği ham veri tablolarına ait alan değerlerinin bileşimlerini içerir. **keep** öğesinden önce kullanılması durumunda, birinci ham veri tablosunun Qlik Sense içinde depolanmadan önce ikinci tabloyla ortak kesişimine azaltılması gerektiğini belirtir.

Aynı adı taşıyan dize fonksiyonunu mu arıyordunuz? Bkz. Right (page 806)

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablename	Yüklenen tablo ile karşılaştırılacak adlandırılmış tablo.
loadstatementveya selectstatement	Yüklenen tablo için **LOAD** veya **SELECT** deyimi.

Örnek

Komut dosyası

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 ];
```

Sonuç

Sonuç tablosu

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

Açıklama

Bu örnek, yalnızca ikinci (sağ) tabloda bulunan değerlerin birleştirildiği Right Join çıktısını gösterir.

Sample

Bir **LOAD** veya **SELECT** deyimine yönelik **sample** öneki, veri kaynağından rastgele sayıda kayıt yüklemek için kullanılır.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
p	0'dan büyük ve 1'den küçük veya buna eşit bir sayı olarak değerlendirilen rastgele seçilmiş ifade. Sayı, belirli bir kaydın okunma olasılığını belirtir. Tüm kayıtlar okunur, ancak yalnızca bazıları Qlik Sense içine yüklenir.

Örnek:

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

Parantezlere izin verilir, ancak gerekli değildir.

Semantic

Kayıtlar arasında ilişki içeren tablolar bir **semantic** önekiyle yüklenebilir. Bu örneğin, bir kaydın bir diğerine işaret ettiği (üst öğe, aittir veya öncel gibi), bir tablo içindeki kendi kendine başvurular olabilir.

Söz Dizimi:

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

Anlamsal yükleme, verilerde navigasyon için kullanılmak üzere filtre bölmelerinde görüntülenebilen anlamsal alanları oluşturur.

semantic deyimi aracılığıyla yüklenen tablolar birleştirilemez.

Örnek:

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

Unless

unless öneki ve soneki bir deyimin veya bir çıkış cümlesinin değerlendirilip değerlendirilmemesi gerektiğini belirleyen koşullu bir cümle oluşturmak için kullanılır. Bu, uzun **if..end if** deyiminin kısa bir alternatifi olarak da görülebilir.

Söz Dizimi:

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

statement veya **exitstatement** ancak **condition** False olarak değerlendirilirse yürütülür.

unless öneki, ek **when** veya **unless** önekleri de dahil olmak üzere, bir veya birden fazla başka deyime zaten sahip olan deyimlerde kullanılabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
statement	Kontrol ifadeleri dışında herhangi bir Qlik Sense kod deyimi.
exitstatement	Bir **exit for**, **exit do** veya **exit sub** cümlesi ya da bir **exit script** deyimi.

Örnekler:

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

When

when öneki ve soneki bir deyimin veya bir çıkış cümlesinin yürütülüp yürütülmemesi gerektiğini belirleyen koşullu bir cümle oluşturmak için kullanılır. Bu, uzun **if..end if** deyiminin kısa bir alternatifi olarak da görülebilir.

Söz Dizimi:

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

statement veya **exitstatement** yalnızca koşul True olarak değerlendirilirse yürütülür.

when öneki, ek **when** veya **unless** önekleri de dahil olmak üzere, bir veya birden fazla başka deyime zaten sahip olan deyimlerde kullanılabilir.

Söz Dizimi:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	True veya False olarak değerlendirilen bir mantıksal ifade.
statement	Kontrol ifadeleri dışında herhangi bir Qlik Sense kod deyimi.
exitstatement	Bir **exit for**, **exit do** veya **exit sub** cümlesi ya da bir **exit script** deyimi.

Example 1:

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

Example 2:

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

Example 3:

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

2.5 Normal kod deyimleri

Normal deyimler genellikle verileri birkaç farklı şekilde işlemek için kullanılır. Bu deyimler kod içinde birçok satıra yazılabilir ve her zaman bir noktalı virgül ";" işaretiyle sonlandırılmalıdır.

Tüm kod anahtar sözcükleri küçük harf ve büyük harften oluşan karakterlerin herhangi bir bileşimiyle yazılabilir. Bununla birlikte, deyimlerde kullanılan alan ve değişken adları büyük/küçük harf duyarlıdır.

Normal kod deyimlerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Alias

alias deyimi, kendisini takip eden kod içinde oluştuğunda yeniden adlandırılacak bir alana göre bir takma isim ayarlamak için kullanılır.

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

Autonumber

Bu deyim, kod yürütme sırasında karşılaşılan bir alandaki her tekil değerlendirilen değer için benzersiz bir tamsayı değeri oluşturur.

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

Binary

binary deyimi, bölüm erişim verisi dahil olmak üzere başka bir QlikView belgesinden verileri yüklemek için kullanılır.

```
Binary  [path] filename
```

comment

Veritabanları ve elektronik tablolardaki alan yorumlarını (meta verileri) görüntülemenin bir yolunu sunar. Uygulamada olmayan alan adı yok sayılır. Bir alan adının birden fazla oluştuğu görülürse, son değer kullanılır.

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

comment table

Veritabanları veya elektronik tablolardaki tablo yorumlarını (meta verileri) görüntülemenin bir yolunu sunar.

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

Connect

Bu işlev Qlik Sense SaaS ürününde mevcut değildir.

CONNECT deyimi, OLE DB/ODBC arabirimi üzerinden bir genel veritabanına Qlik Sense erişimi tanımlamak için kullanılır. ODBC için, veri kaynağı ilk olarak ODBC yöneticisi kullanılarak belirlenmelidir.

```
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

Declare deyimi, alanlar veya fonksiyonlar arasındaki ilişkileri tanımlayabileceğiniz alan tanımları oluşturmak için kullanılır. Boyut olarak kullanılabilecek türetilmiş alanları otomatik olarak oluşturmak için bir alan tanımları kümesi kullanabilir. Örneğin, bir takvim tanımı oluşturabilir ve bunu kullanarak bir tarih alanından yıl, ay, hafta ve gün gibi ilgili boyutları oluşturabilirsiniz.

```
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

Derive deyimi, **Declare** deyimi ile oluşturulan bir alan tanımını temel alan türetilmiş alanlar oluşturmak için kullanılır. Hangi alanlar için verilerin türetileceğini belirtebilir veya bunları alan etiketlerine göre açık ya da örtük bir şekilde türetebilirsiniz.

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

Direct Query

DIRECT QUERY deyimi, ODBC veya OLE DB bağlantısı aracılığıyla ve Direct Discovery işlevini kullanarak tablolara erişmenize izin verir.

```
Direct Query [path]
```

Directory

Directory deyimi, yeni bir **Directory** deyimi oluşturulana dek sonraki **LOAD** deyimlerinde hangi dizinde veri dosyaları aranacağını belirler.

```
Directory [path]
```

Disconnect

Disconnect deyimi geçerli ODBC/OLE DB/Özel bağlantısını sonlandırır. Bu deyim isteğe bağlıdır.

```
Disconnect
```

drop field

Bir veya daha fazla Qlik Sense alanı kod yürütmesi sırasında istenildiği zaman veri modelinden ve dolayısıyla bellekten **drop field** deyimi aracılığıyla bırakılabilir.

> *Hem **drop field** hem de **drop fields** etkileri açısından aralarında fark olmayan ve izin verilen biçimlerdir. Herhangi bir tablo belirtilmemişse, alan oluştuğu tüm tablolara bırakılır.*

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

drop table

Bir veya daha fazla Qlik Sense dahili tablosu kod yürütmesi sırasında istenildiği zaman veri modelinden ve dolayısıyla bellekten **drop table** deyimi aracılığıyla bırakılabilir.

> ***drop table** ve **drop tables** biçimlerinin ikisi de kabul edilir.*

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

Execute

Execute deyimi, Qlik Sense verileri yüklediği sırada diğer programları çalıştırmak için kullanılır. Örneğin, gerekli olan dönüştürmeleri yapmak için.

```
Execute commandline
```

FlushLog

FlushLog deyimi, Qlik Sense uygulamasını kod belleğinin içeriğini kod günlük dosyasına yazmaya zorlar.

```
FlushLog
```

Force

force deyimi, Qlik Sense uygulamasını alan değerlerini ve kendisinden sonra gelen **LOAD** ve **SELECT** deyimlerinin alan değerlerini yalnızca büyük harflerle, yalnızca küçük harflerle, her zaman ilk harfi büyük olarak veya göründüğü gibi (karışık) yorumlamaya zorlar. Bu deyim, tablodan alınan alan değerlerinin farklı kurallara göre ilişkilendirilmesini mümkün kılar.

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

LOAD

LOAD deyimi, alanları bir dosyadan, kod içinde tanımlanmış verilerden, daha önceden yüklenmiş tablodan, web sayfasından, ardından gelen **SELECT** deyiminin sonucundan veya verileri otomatik olarak oluşturarak yükler. Analiz bağlantılarından da veriler yüklenebilir.

```
 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

let deyimi **set** deyiminin tamamlayıcısıdır ve kod değişkenlerini tanımlamak için kullanılır. **let** deyimi, **set** deyiminin aksine "=" işaretinin sağındaki ifadeyi, kodun çalışma zamanında değişkene atanmadan önce değerlendirir.

```
Let variablename=expression
```

Loosen Table

Bir veya daha fazla Qlik Sense dahili veri tablosu, **Loosen Table** deyimi kullanılarak kod yürütmesi sırasında açık şekilde gevşek bağlı olarak bildirilebilir. Bir tablo gevşek bağlı olduğunda, tabloda bulunan alan değerleri arasındaki tüm ilişkiler kaldırılır. Benzer bir etki, gevşek bağlı tablonun her bir alanının bağımsız, ilişkisiz tablolar olarak yüklenmesiyle elde edilebilir. Gevşek bağlı özelliği, test sırasında veri yapısının farklı bölümlerinin geçici olarak ayrı tutulmasında yararlı olabilir. Gevşek bağlı bir tablo, tablo görüntüleyicisinde noktalı çizgilerle gösterilebilir. Kod içerisinde bir veya daha fazla **Loosen Table** deyimi

kullanılması, Qlik Sense uygulamasının kodu yürütmeden önce gevşek bağlı tablolarda yapılan ayarları göz ardı etmesine neden olur.

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

Map ... using

map ... using deyimi, belirli bir alan değerini veya ifadesini belirli bir eşleme tablosunun değerlerine eşlemek için kullanılır. Eşleme tablosu **Mapping** deyimi aracılığıyla oluşturulur.

```
Map  *fieldlist Using  mapname
```

NullAsNull

NullAsNull deyimi, NULL değerlerin **NullAsValue** deyimi tarafından daha önce ayarlanmış dize değerlerine dönüştürülmesi işlemini kapatır.

```
NullAsNull *fieldlist
```

NullAsValue

NullAsValue deyimi, hangi alanlar için NULL öğesinin bir değere döndürülmesi gerektiğini belirtir.

```
NullAsValue *fieldlist
```

Qualify

Qualify deyimi, alan adlarının nitelendirilmesi, yani alan adlarının tablo adıyla aynı öneki alması özelliğini açmak için kullanılır.

```
Qualify *fieldlist
```

Rem

rem deyimi, koda açıklama veya yorum eklemek veya kod deyimlerini kaldırmadan geçici olarak etkinliklerini kaldırmak için kullanılır.

```
Rem string
```

Rename Field

Bu kod fonksiyonu, bir veya daha fazla var olan Qlik Sense alanını yüklendikten sonra yeniden adlandırır.

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

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

Rename Table

Bu kod fonksiyonu, bir veya daha fazla var olan Qlik Sense dahili tabloyu yüklendikten sonra yeniden adlandırır.

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

Section

section deyimiyle, sonraki **LOAD** ve **SELECT** deyimlerinin veri veya erişim haklarının bir tanımı olarak ele alınmasına ilişkin seçimi tanımlamak mümkündür.

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

Select

Bir ODBC veri kaynağından veya bir OLE DB sağlayıcısından alanların seçilmesi, standart SQL **SELECT** deyimleriyle gerçekleştirilir. Bununla birlikte, **SELECT** deyimlerinin kabul edilip edilmemesi, kullanılan ODBC sürücüsüne veya OLE DB sağlayıcısına bağlıdır.

```
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

set deyimi kod değişkenlerini tanımlamak için kullanılır. Bunlar dizelerin, yolların, sürücülerin ve benzeri öğelerin yerini alması için kullanılabilir.

```
Set variablename=string
```

Sleep

sleep deyimi kod yürütmesini belirtilen süre kadar duraklatır.

```
Sleep n
```

SQL

SQL deyimi, bir ODBC veya OLE DB bağlantısı aracılığıyla rastgele bir SQL komutu göndermenize olanak tanır.

```
SQL sql_command
```

SQLColumns

sqlcolumns deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının sütunlarını açıklayan bir alan setini döndürür.

```
SQLColumns
```

SQLTables

sqltables deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının tablolarını açıklayan bir alan setini döndürür.

```
SQLTables
```

SQLTypes

sqltypes deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının türlerini açıklayan bir alan setini döndürür.

```
SQLTypes
```

Star

Veritabanındaki bir alanın tüm değerler kümesini temsilen kullanılan dize **star** deyimi aracılığıyla ayarlanabilir. Sonrasında gelen **LOAD** ve **SELECT** deyimlerini etkiler.

```
Star is [ string ]
```

Store

Store deyimi bir QVD, CSV veya text dosyası olşuşturur.

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

Tag

Bu kod deyimi, bir veya daha fazla alana veya tabloya etiket atama yolu sağlar. Uygulamada mevcut olmayan bir alanı veya tabloyu etiketleme girişimi olursa etiketleme yoksayılacaktır. Bir alan veya etiket adının çakışan oluşları varsa, son değer kullanılır.

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

Trace

trace deyimi, kullanıldığında, **Kod Yürütme İlerlemesi** penceresine ve kod günlük dosyasına bir dize yazar. Bu deyim, hata ayıklama amaçlı kullanımda çok faydalıdır. **trace** deyimi öncesinde hesaplanan değişkenlerin $ genişletmelerini kullanarak, mesajı özelleştirebilirsiniz.

```
Trace string
```

Unmap

Unmap deyimi, arkasından gelen yüklenmiş alanlar için olan önceki bir **Map … Using** deyimi ile belirlenen alan değeri eşlemesini devre dışı bırakır.

```
Unmap *fieldlist
```

Unqualify

Unqualify deyimi, daha önce **Qualify** deyimiyle açılmış olan alan adlarının nitelenmesini kapatmak için kullanılır.

```
Unqualify *fieldlist
```

Untag

Bu kod deyimi, alan veya tablolardan etiket kaldırma yolu sağlar. Uygulamada mevcut olmayan bir alandan veya tablodan etiket kaldırma girişimi olursa etiket kaldırma yoksayılacaktır.

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

```
Tag table tablelist with tagname
```

Alias

alias deyimi, kendisini takip eden kod içinde oluştuğunda yeniden adlandırılacak bir alana göre bir takma isim ayarlamak için kullanılır.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fieldname	Kaynak verilerinizdeki alanın adı
aliasname	Yerine kullanmak istediğiniz bir takma isim

Örnekler ve sonuçlar:

Örnek	Sonuç
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Bu ifadeyle tanımlanan ad değişiklikleri, sonrasında gelen tüm **SELECT** ve **LOAD** deyimleri üzerinde kullanılır. Bir alan adı için yeni bir takma isim, kod içinde kendisinden sonra gelen bir konumda yeni bir **alias** deyimiyle tanımlanabilir.

AutoNumber

Bu deyim, kod yürütme sırasında karşılaşılan bir alandaki her tekil değerlendirilen değer için benzersiz bir tamsayı değeri oluşturur.

Ayrıca **LOAD** deyimi içinde *autonumber (page 438)* fonksiyonunu kullanabilirsiniz ancak optimize yükleme kullanmak istediğinize bunun bazı sınırlandırmaları vardır. Verileri önce **QVD** dosyasından yükleyerek ve ardından değerleri simge anahtarlarına dönüştürmek üzere **AutoNumber** deyimini kullanarak bir optimize yükleme oluşturabilirsiniz.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	Virgülle ayrılmış alan listesi; burada değerler benzersiz bir tamsayı değeriyle değiştirilmelidir. Eşleşen adlara sahip tüm alanları dahil etmek için ? ve * joker karakterlerini kullanabilirsiniz. Ayrıca tüm alanları dahil etmek için * simgesini de kullanabilirsiniz. Jokerler kullanılırken alan adlarını tırnak içine almanız gerekir.
namespace	`namespace` **`kullanımı`** isteğe bağlıdır. Farklı alanlarda aynı değerlerin aynı anahtara sahip olduğu bir namespace oluşturmak için bu seçeneği kullanabilirsiniz. Bu seçeneği kullanmazsanız tüm alanların ayrı bir anahtar dizini olur.

Sınırlamalar:

Kodunuzda birkaç **LOAD** deyimi olduğunda **AutoNumber** deyimini son **LOAD** deyiminden sonra koymanız gerekir.

Örnek - AutoNumber'lı kod

Kod örneği

Bu örnekte, veriler önce **AutoNumber** deyimi olmadan yüklenir. Daha sonra **AutoNumber** deyimi etkisini göstermek için eklenir.

Örnekte kullanılan veriler

Aşağıdaki kod örneğini oluşturmak için veri yükleme düzenleyicisinde aşağıdaki verileri satır içi yükleme olarak yükleyin. **AutoNumber** deyimini şimdilik derleme dışı bırakın.

```
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;
```

Görselleştirme oluşturma

Qlik Sense sayfasında iki tablo görselleştirmesi oluşturun. **KeyToOtherTable**, **Region**, **Year**, **Month** ve **Sales** alanlarını boyut olarak ilk tabloya ekleyin. **KeyToOtherTable**, **Region**, **Year**, **Month** ve **Budget** alanlarını boyut olarak ikinci tabloya ekleyin.

Sonuç

RegionSales tablosu

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

Budget tablosu

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

Açıklama

Örnek, iki tabloyu birbirine bağlayan bileşik **KeyToOtherTable** alanını göstermektedir. **AutoNumber** kullanılmamaktadır. **KeyToOtherTable** değerlerinin uzunluğuna dikkat edin.

AutoNumber deyimini ekleme

Komut dosyasında **AutoNumber** deyimini derleme dışı bırakın.

```
AutoNumber KeyToOtherTable;
```

Sonuç

RegionSales tablosu

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

Budget tablosu

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

Açıklama

KeyToOtherTable alanının değerleri benzersiz tamsayı değerleriyle değiştirilmiştir ve bunun sonucunda alan değerleri kısaltılarak bellekten tasarruf sağlanmıştır. Her iki tablodaki anahtar alanları **AutoNumber**'dan etkilenir ve tablolar birbirine bağlı kalır. Örnek, gösterim amacına yönelik olarak kısa tutulmuştur, ancak çok sayıda satır içeren bir tablo ile daha anlamlı olacaktır.

Binary

binary deyimi, bölüm erişim verisi dahil olmak üzere başka bir Qlik Sense uygulaması veya QlikView belgesinden verileri yüklemek için kullanılır. Uygulamanın sayfalar, hikayeler, görselleştirmeler, ana öğeler veya değişkenler gibi diğer öğeleri dahil edilmez.

Kodda yalnızca bir **binary** deyimine izin verilir. **binary** deyimi kodun ilk deyimi olmalıdır. Genellikle kodun başında yer alan SET deyimlerinin bile önüne gelir.

Söz Dizimi:

```
binary [path] filename
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
path	Klasör veri bağlantısının referansı olması gereken dosya yolu. Bu, dosya Qlik Sense çalışma dizininde yer almıyorsa gereklidir. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Bu kod satırını içeren uygulamaya göreceli. **Örnek:** *data*
filename	.qvw veya .qvf dosya uzantısı da dahil olmak üzere dosyanın adı.

Sınırlamalar:

binary deyimini uygulama kimliğine başvurarak aynı Qlik Sense Enterprise dağıtımındaki bir uygulamadan veri yüklemek için kullanamazsınız. Yalnızca *.qvf* dosyasından yükleme gerçekleştirebilirsiniz.

Örnekler

Dize	Açıklama
`Binary lib://DataFolder/customer.qvw;`	Bu örnekte dosya, **Klasör** veri bağlantısında yer almalıdır. Bu, örneğin, yöneticinizin Qlik Sense sunucusunda oluşturduğu bir klasör olabilir. Veri yükleme düzenleyicisinde **Yeni bağlantı oluştur**'a tıklayıp **Dosya konumları** bölümünde **Klasör**'ü seçin.
`Binary customer.qvf;`	Bu örnekte, dosya Qlik Sense çalışma dizininde olmalıdır.
`Binary c:\qv\customer.qvw;`	Mutlak dosya yolu kullanan bu örnek, yalnızca eski kod oluşturma modunda çalışacaktır.

Comment field

Veritabanları ve elektronik tablolardaki alan yorumlarını (meta verileri) görüntülemenin bir yolunu sunar. Uygulamada olmayan alan adı yok sayılır. Bir alan adının birden fazla oluştuğu görülürse, son değer kullanılır.

Söz Dizimi:

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

Kullanılan eşleme tablosu birincisi alan adlarını ve ikincisi yorumları içeren iki sütuna sahip olmalıdır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
**fieldlist*	Yorum yapılacak alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.
mapname	Bir eşleme **LOAD** veya eşleme **SELECT** deyiminde daha önce okunmuş bir eşleme tablosunun adı.
fieldname	Yorum yapılması gereken alanın adı.
comment	Alana eklenmesi gereken yorum.

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

Veritabanları veya elektronik tablolardaki tablo yorumlarını (meta verileri) görüntülemenin bir yolunu sunar.

Uygulamada olmayan tablo adları yok sayılır. Bir tablo adının birden fazla oluşumuna rastlanırsa son değer kullanılır. Bir veri kaynağından yorumları okumak için anahtar sözcük kullanılabilir.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
tablelist	(table{,table})
mapname	Bir eşleme **LOAD** veya eşleme **SELECT** deyiminde daha önce okunmuş bir eşleme tablosunun adı.
tablename	Yorum yapılması gereken tablonun adı.
comment	Tabloya eklenmesi gereken yorum.

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

CONNECT deyimi, OLE DB/ODBC arabirimi üzerinden bir genel veritabanına Qlik Sense erişimi tanımlamak için kullanılır. ODBC için, veri kaynağı ilk olarak ODBC yöneticisi kullanılarak belirlenmelidir.

Bu işlev Qlik Sense SaaS ürününde mevcut değildir.

Bu deyim, yalnızca standart modda klasör veri bağlantılarını destekler.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Bağlantı dizgesi, veri kaynağı adı ve bir veya daha fazla bağlantı teknik özelliği öğelerinin isteğe bağlı listesidir. Veri kaynağı adı boşluk içerirse veya herhangi bir bağlantı teknik özelliği öğesi listelenirse, bağlantı dizgesi tırnak işaretleri içine alınmalıdır.
	datasourcename, tanımlı bir ODBC veri kaynağı veya OLE DB sağlayıcısını tanımlayan bir dize olmalıdır.
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	Olası bağlantı teknik özelliği öğeleri farklı veritabanları arasında farklılık gösterebilir. Bazı veritabanları için, yukarıdakilerden farklı başka öğeler de olasıdır. OLE DB için, bağlantıya özel öğelerin bazıları zorunludur ve isteğe bağlı değildir.

connection	Veri bağlantısının adı veri yükleme düzenleyicisinde depolanır.

ODBC, **CONNECT** öncesine yerleştirilirse ODBC arabirimi kullanılır; aksi takdirde OLE DB kullanılır.

LIB CONNECT TO kullanılırsa, veri yükleme düzenleyicisinde oluşturulmuş bir depolanan veri bağlantısı kullanılarak veritabanına bağlanılır.

Example 1:

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

Bu deyim aracığıyla tanımlanan veri kaynağı, yeni bir **CONNECT** deyimi yapılana kadar, sonraki **Select (SQL)** deyimleri tarafından kullanılır.

Example 2:

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

Connect32

Bu deyim **CONNECT** deyimiyle aynı şekilde kullanılır, ancak 64 bit sistemi 32 bit ODBC/OLE DB sağlayıcısı kullanmaya zorlar. Özel bağlantılar için uygulanabilir değildir.

Connect64

Bu deyim **CONNECT** deyimiyle aynı şekilde kullanılır, ancak 64 bit sağlayıcı kullanımını zorlar. Özel bağlantılar için uygulanabilir değildir.

Declare

Declare deyimi, alanlar veya fonksiyonlar arasındaki ilişkileri tanımlayabileceğiniz alan tanımları oluşturmak için kullanılır. Boyut olarak kullanılabilecek türetilmiş alanları otomatik olarak oluşturmak için bir alan tanımları kümesi kullanabilir. Örneğin, bir takvim tanımı oluşturabilir ve bunu kullanarak bir tarih alanından yıl, ay, hafta ve gün gibi ilgili boyutları oluşturabilirsiniz.

Yeni bir alan tanımı ayarlamak veya mevcut bir tanıma göre alan tanımı oluşturmak için **Declare** seçeneğini kullanabilirsiniz.

Yeni bir alan tanımı ayarlama

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
definition_name	İki nokta üst üste ile biten alan tanımının adı. *Alan tanımlarının adı olarak autoCalendar'ı kullanmayın; çünkü bu ad, otomatik olarak oluşturulan takvim şablonları için ayrılmıştır.* **Örnek:** `Calendar:`
tag_list	Alan tanımından türetilen alanlara uygulanacak etiketlerin virgülle ayrılmış listesi. Etiketlerin uygulanması isteğe bağlıdır ancak $date, $numeric veya $text gibi sıralama düzenini belirtmek için kullanılan etiketleri uygulamazsanız, türetilen alan varsayılan olarak yükleme düzenine göre sıralanacaktır. **Örnek:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`

Bağımsız Değişken	Açıklama
parameter_list	Parametrelerin virgülle ayrılmış listesi. `name=value` biçiminde bir parametre tanımlanır ve alan tanımı yeniden kullanıldığında geçersiz kılınabilecek bir başlangıç değeri atanır. İsteğe bağlı. **Örnek:** `first_month_of_year = 1`
field_list	Alanlar için alan tanımı kullanıldığında oluşturulacak virgülle ayrılmış bir liste. `<expression> As field_name tagged tag` biçiminde bir alan tanımlanır. Türetilen alanların oluşturulması gereken veri alanına referansta bulunmak için `$1` öğesini kullanın. **Örnek:** `Year($1) As Year tagged ('$numeric')`

Örnek:

```
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')
;
```

Takvim artık tanımlanmıştır ve bunu yüklenen tarih alanlarına uygulayabilirsiniz (bu durumda **Derive** cümlesini kullanan OrderDate ve ShippingDate).

Mevcut alan tanımını yeniden kullanma

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
definition_name	İki nokta üst üste ile biten alan tanımının adı. **Örnek:** `MyCalendar:`
existing_definition	Yeni alan tanımı oluşturulurken yeniden kullanılacak alan tanımı. Alan ifadelerinde kullanılan değeri değiştirmek için parameter_assignment kullanmanız hariç olmak üzere, yeni alan tanımı temel aldığı tanımla aynı işlevi görecektir. **Örnek:** `Using Calendar`
parameter_assignment	Parametre atamalarının virgülle ayrılmış listesi. `name=value` biçiminde bir parametre ataması tanımlanır ve temel alan tanımında ayarlanan parametre değerini geçersiz kılar. İsteğe bağlı. **Örnek:** `first_month_of_year = 4`

Örnek:

Bu örnekte, önceki örnekte oluşturulan takvim tanımını yeniden kullanıyoruz. Bu durumda, Nisan ayında başlayan bir mali yıl kullanmak istiyoruz. Bu, 4 değeri first_month_of_year parametresine atanarak elde edilir; bu durumda tanımlanan DayNumberOfYear alanı etkilenir.

Örnek, önceki örnekte bulunan örnek veri ve alan tanımını kullandığınızı varsayar.

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

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

Veri kodunu yeniden yüklediğinizde, oluşturulan alanlar OrderDate.MyCalendar.* ve ShippingDate.MyCalendar.* adlarıyla sayfa düzenleyicisinde kullanılabilir.

Derive

Derive deyimi, **Declare** deyimi ile oluşturulan bir alan tanımını temel alan türetilmiş alanlar oluşturmak için kullanılır. Hangi alanlar için verilerin türetileceğini belirtebilir veya bunları alan etiketlerine göre açık ya da örtük bir şekilde türetebilirsiniz.

Söz Dizimi:

```
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
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
definition	Alanlar türetilirken kullanılacak alan tanımının adı. **Örnek:** `Calendar`
field_list	Alan tanımına göre türetilen alanların oluşturulması gereken veri alanlarının virgülle ayrılmış listesi. Veri alanları, kodda daha önce yüklediğiniz alanlar olmalıdır. **Örnek:** `OrderDate, ShippingDate`
tag_list	Etiketlerin virgülle ayrılmış listesi. Türetilen alanlar, tüm veri alanları için listelenen etiketlerin herhangi biriyle oluşturulacaktır. Etiket listesi, yuvarlak ayraç içine alınmalıdır. **Örnek:** `('$date','$timestamp')`

Örnekler:

- Belirli veri alanları için alanlar türetin.
  Bu durumda OrderDate ve ShippingDate alanlarını belirtiriz.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Belirli bir etiketle tüm alanlar için alanlar türetin.
  Bu durumda, `$date` etiketi olan tüm alanlar için Calendar öğesine dayanan alanlar türetiriz.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Alan tanımı etiketiyle tüm alanlar için alanlar türetin.
  Bu durumda, Calendar alan tanımıyla aynı etikete (bu durumda `$date`) sahip tüm veri alanları için alanlar türetiriz.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

DIRECT QUERY deyimi, ODBC veya OLE DB bağlantısı aracılığıyla ve Direct Discovery işlevini kullanarak tablolara erişmenize izin verir.

Söz Dizimi:

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

DIMENSION, **MEASURE** ve **DETAIL** anahtar sözcükleri istenen sırada kullanılabilir.

DIMENSION ve **FROM** anahtar sözcük cümleleri tüm **DIRECT QUERY** deyimlerinde gereklidir. **FROM** anahtar sözcüğü **DIMENSION** anahtar sözcüğünden sonra görünmelidir.

DIMENSION anahtar sözcüğünden hemen sonra belirtilen alanlar belleğe yüklenir ve bellek içi ile Direct Discovery verileri arasında ilişkiler oluşturmak için kullanılabilir.

> *DIRECT QUERY deyimi DISTINCT veya GROUP BY cümlelerini içeremez.*

MEASURE anahtar sözcüğünü kullanarak Qlik Sense uygulamasının "meta düzeyinde" farkında olduğu alanlar tanımlayabilirsiniz. Bir hesaplama alanının gerçek verileri, veri yükleme işlemi sırasında yalnızca veritabanında bulunur ve bir görselleştirmede kullanılan grafik ifadelerinin yönlendirmesiyle "amaca özel" (ad hoc) esasına göre getirilir.

Genellikle, boyut olarak kullanılacak ayrık değerli alanların **DIMENSION** anahtar sözcüğüyle yüklenmesi gerekirken, toplamalarda kullanılacak sayıların yalnızca **MEASURE** anahtar sözcüğüyle seçilmesi gerekir.

DETAIL alanları kullanıcının detaya inme tablo kutusunda görüntülemek isteyebileceği bilgileri veya ayrıntıları (yorum alanları gibi) sağlar. **DETAIL** alanları grafik ifadelerinde kullanılamaz.

Tasarım olarak, **DIRECT QUERY** deyimi, SQL desteği sağlayan veri kaynakları için veri kaynağı açısından tarafsızdır. Bu nedenle, aynı **DIRECT QUERY** deyimi, değişiklik olmadan farklı SQL veritabanları için kullanılabilir. Direct Discovery, veritabanına uygun sorguları gerektiği gibi oluşturur.

Kullanıcı sorgulanacak veritabanını bildiğinde ve SQL'e yönelik veritabanına özgü uzantıların açıklarından yararlanmak istediğinde yerel veri kaynağı söz dizimi kullanılabilir. Yerel veri kaynağı söz dizimi şu şekilde desteklenir:

- **DIMENSION** ve **MEASURE** cümlelerinde alan ifadeleri olarak
- **WHERE** cümlesinin içeriği olarak

Örnekler:

```
DIRECT QUERY
      DIMENSION Dim1, Dim2

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

DIRECT QUERY
      DIMENSION Dim1, Dim2

      MEASURE X, Y

      FROM TableName

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

> *Şu terimler anahtar sözcük olarak kullanılır ve bu nedenle alıntılanmadan sütun veya alan adları olarak kullanılamaz: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where*

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
fieldlist	Alan teknik özelliklerinin virgülle ayrılmış listesi, *fieldname {, fieldname}*. Alan teknik özelliği bir alan adı olabilir; bu durumda veritabanı sütun adı ve Qlik Sense alan adı için aynı ad kullanılır. Veya alan teknik özelliği bir "alan takma ismi" olabilir; bu durumda veritabanı ifadesine veya sütun adına bir Qlik Sense alan adı verilir.
tablelist	Verilerin yükleneceği veritabanındaki tablo veya görünüm adlarının bir listesi. Genellikle, veritabanında gerçekleştirilen JOIN'i içeren bir görünümdür.

where_ clause	Veritabanı **WHERE** cümlelerinin tam söz dizimi burada tanımlanmamaktadır; ancak çoğu SQL "ilişkisel ifadesine" izin verilir; fonksiyon çağrıları, dizeler için **LIKE** işleci, **IS NULL** ile **IS NOT NULL** ve **IN. BETWEEN** kullanımları da buna dahildir.
	
	NOT, belirli anahtar sözcükler üzerindeki değiştiricinin aksine birli işleçtir.
	
	Örnekler:
	
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	
	`WHERE NOT X in (1,2,3)`
	Son örnek aşağıdaki gibi yazılamaz:
	
	`WHERE X NOT in (1,2,3)`

Örnek:

Bu örnekte Dim1, Dim2, Num1, Num2 ve Num3 alanlarını içeren TableName adında bir veritabanı tablosu kullanılmaktadır.Dim1 ve Dim2, Qlik Sense veri kümesine yüklenir

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

Dim1 ve Dim2 boyut olarak kullanıma açık olacaktır. Num1, Num2 ve Num3 toplamalar için kullanılabilecektir. Dim1 ve Dim2 de toplamalar için kullanılabilecektir. Dim1 ve Dim2 öğelerinin kullanılabildiği toplamaların türü bunların veri türlerine göre değişir. Örneğin, birçok durumda **DIMENSION** alanları adlar veya hesap numaraları gibi dize verilerini içerir. Bu alanlar toplanamaz, ancak sayılabilir: `count(Dim1)`.

__DIRECT QUERY__ deyimleri doğrudan kod düzenleyicisinde yazılır. __DIRECT QUERY__ deyimlerinin oluşturulmasını basitleştirmek amacıyla bir veri bağlantısından __SELECT__ deyimi oluşturabilir ve ardından, oluşturulan kodu düzenleyerek bir __DIRECT QUERY__ deyimine dönüştürebilirsiniz.
Örneğin, __SELECT__ deyimi:

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

şu __DIRECT QUERY__ deyimine dönüştürülebilir:

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direct Discovery alan listeleri

Alan listesi, alan teknik özelliklerinin virgülle ayrılmış listesidir (*fieldname {, fieldname}*). Alan teknik özelliği bir alan adı olabilir; bu durumda veritabanı sütun adı ve alan adı için aynı ad kullanılır. Veya alan teknik özelliği bir alan takma ismi olabilir; bu durumda veritabanı ifadesine veya sütun adına bir Qlik Sense alan adı verilir.

Alan adları, basit adlar veya alıntılanan adlar olabilir. Basit ad, alfabetik Unicode karakteriyle başlar ve bu karakteri alfabetik veya sayısal karakterlerden veya alt çizgilerden oluşan bir kombinasyon takip eder. Alıntılanan adlar çift tırnak işaretiyle başlar ve bir karakter dizisini içerir. Alıntılanan ad çift tırnak işareti içeriyorsa, bu tırnak işaretleri birbirine bitişik iki çift tırnak işareti kullanılarak gösterilir.

Qlik Sense alan adları büyük/küçük harf duyarlıdır. Veritabanı alan adları, veritabanına bağlı olarak büyük/küçük harf duyarlı olabilir veya olmayabilir. Direct Discovery sorgusu tüm alan tanımlayıcılarının ve takma isimlerinin büyük/küçük harf durumunu korur. Aşağıdaki örnekte "MyState" takma ismi, "STATEID" adlı veritabanı sütunundan verileri depolamak üzere dahili olarak kullanılır.

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

Bunun sonucu, takma isim içeren bir **SQL Select** deyiminin sonucundan farklıdır. Takma isim açıkça alıntılanmazsa sonuç, hedef veritabanının döndürdüğü sütunun varsayılan büyük/küçük harf durumunu içerir. Aşağıdaki örnekte Oracle veritabanına yönelik bir **SQL Select** deyimi, takma isim büyük/küçük harf karışık olarak belirtilmesine karşın, dahili Qlik Sense takma ismi olarak tümü büyük harflerle yazılan "MYSTATE," öğesini oluşturur. **SQL Select** deyimi veritabanı tarafından döndürülen sütun adını kullanır ve bu durumda Oracle için tümü büyük harf olur.

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

Bu davranıştan kaçınmak amacıyla takma ismi belirtmek için LOAD deyimini kullanın.

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

Bu örnekte, "STATEID" sütunu dahili olarak Qlik Sense tarafından "MyState" şeklinde depolanır.

Veritabanı skaler ifadelerinin çoğuna alan teknik özelliği olarak izin verilir. Fonksiyon çağrıları da alan teknik özellikleri olarak kullanılabilir. İfadeler tekli tırnak işaretleri içinde içerilen boole, sayısal veya dizeler olan sabitleri içerebilir (eklenmiş tekli tırnak işaretleri birbirine bitişik tekli tırnak işaretleriyle gösterilir).

Örnekler:

```
DIRECT QUERY

   DIMENSION

      SalesOrderID, RevisionNumber

   MEASURE

      SubTotal AS "Sub Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      "SalesOrderID" AS "Sales Order ID"

   MEASURE

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

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

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

```
            Molecules/6.02e23 AS Moles

         MEASURE

            Num1 AS numA

      FROM TableName;


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

Direct Discovery, **LOAD** deyimlerinde toplamaları kullanmayı desteklemez. Toplamalar kullanılırsa sonuçlar öngörülemez olur. Aşağıdaki gibi bir **LOAD** deyimi kullanılmamalıdır:

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

SUM öğesi **LOAD** deyimi içinde olmamalıdır.

Direct Discovery, **Direct Query** deyimlerinde Qlik Sense fonksiyonlarını da desteklemez. Örneğin, **DIMENSION** alanı için aşağıdaki teknik özellik, "Mth" alanının bir görselleştirmede boyut olarak kullanılması halinde hatayla sonuçlanır:

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

Directory

Directory deyimi, yeni bir **Directory** deyimi oluşturulana dek sonraki **LOAD** deyimlerinde hangi dizinde veri dosyaları aranacağını belirler.

Söz Dizimi:

```
Directory[path]
```

Directory deyimi bir **path** olmadan kullanılır veya unutulursa Qlik Sense, Qlik Sense çalışma dizinine bakar.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
path	data dosyasının yolu olarak yorumlanabilecek bir metin. Yol, dosyanın yoludur ve şunlardan biri olabilir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data* • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler:

```
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

Disconnect deyimi geçerli ODBC/OLE DB/Özel bağlantısını sonlandırır. Bu deyim isteğe bağlıdır.

Söz Dizimi:

```
Disconnect
```

Yeni bir **connect** deyimi yürütüldüğünde veya kod yürütmesi bittiğinde bağlantı otomatik olarak sona erdirilir.

Örnek:

```
Disconnect;
```

Drop

Drop kod anahtar sözcüğü, veritabanından alınan tabloları veya alanları bırakmak için kullanılabilir.

Drop field

Bir veya daha fazla Qlik Sense alanı kod yürütmesi sırasında istenildiği zaman veri modelinden ve dolayısıyla bellekten **drop field** deyimi aracılığıyla bırakılabilir.

*Hem **drop field** hem de **drop fields** etkileri açısından aralarında fark olmayan ve izin verilen biçimlerdir. Herhangi bir tablo belirtilmemişse, alan oluştuğu tüm tablolara bırakılır.*

Söz Dizimi:

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

Örnekler:

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

Drop table

Bir veya daha fazla Qlik Sense dahili tablosu kod yürütmesi sırasında istenildiği zaman veri modelinden ve dolayısıyla bellekten **drop table** deyimi aracılığıyla bırakılabilir.

Söz Dizimi:

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

***drop table** ve **drop tables** biçimlerinin ikisi de kabul edilir.*

Aşağıdaki öğeler bunun sonucu olarak kaybolur:

- Gerçek tablolar.
- Geriye kalan tabloların parçası olmayan tüm alanlar.
- Özel olarak bırakılan tablolardan gelen geriye kalan alanlardaki alan değerleri.

Örnekler ve sonuçlar:

Örnek	Sonuç
`drop table Orders, Salesmen, T456a;`	Bu satır bellekten üç tablonun bırakılmasına yol açar.

`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;`
	Tab2 tablosu oluşturulduktan sonra, *Tab1* tablosu bırakılır.

Drop table

Bir veya daha fazla Qlik Sense dahili tablosu kod yürütmesi sırasında istenildiği zaman veri modelinden ve dolayısıyla bellekten **drop table** deyimi aracılığıyla bırakılabilir.

Söz Dizimi:

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

> ***drop table*** *ve* ***drop tables*** *biçimlerinin ikisi de kabul edilir.*

Aşağıdaki öğeler bunun sonucu olarak kaybolur:

- Gerçek tablolar.
- Geriye kalan tabloların parçası olmayan tüm alanlar.
- Özel olarak bırakılan tablolardan gelen geriye kalan alanlardaki alan değerleri.

Örnekler ve sonuçlar:

Örnek	Sonuç
`drop table Orders, Salesmen, T456a;`	Bu satır bellekten üç tablonun bırakılmasına yol açar.

Örnek	Sonuç
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;	*Tab2* tablosu oluşturulduktan sonra, *Tab1* tablosu bırakılır.

Execute

Execute deyimi, Qlik Sense verileri yüklediği sırada diğer programları çalıştırmak için kullanılır. Örneğin, gerekli olan dönüştürmeleri yapmak için.

Bu işlev Qlik Sense SaaS ürününde mevcut değildir.

Bu deyim, standart modda desteklenmez.

Söz Dizimi:

```
execute commandline
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
commandline	İşletim sistemi tarafından komut satırı olarak yorumlanabilen bir metin. Mutlak dosya yollarına veya lib:// klasör yoluna referansta bulunabilirsiniz.

Execute öğesini kullanmak isterseniz aşağıdaki koşulların karşılanması gerekir:

- Eski modda çalışmanız gerekir (Qlik Sense ve Qlik Sense Desktop için geçerlidir).
- OverrideScriptSecurity öğesini *Settings.ini*'de 1 olarak ayarlamanız gerekir (Qlik Sense için geçerlidir).
  Settings.ini, *C:\ProgramData\Qlik\Sense\Engine* yolunda yer alır ve genellikle boş bir dosyadır.

*OverrideScriptSecurity öğesini **Execute** etkinleştirilecek şekilde ayarlarsanız, tüm kullanıcılar sunucuda dosyaları yürütülebilir. Örneğin, bir kullanıcı uygulamaya yürütülebilir dosya ekleyebilir ve dosyayı veri kod dosyasında yürütebilir.*

Aşağıdakileri yapın:

1. *Settings.ini*'nin kopyasını oluşturun ve metin düzenleyicisinde açın.
2. Dosyanın ilk satırda *[Ayarlar 7]*'yi içerdiğini kontrol edin.
3. Yeni bir satır ekleyin ve *OverrideScriptSecurity=1* yazın.
4. Dosyanın sonuna boş bir satır ekleyin.
5. Dosyayı kaydedin.
6. *Settings.ini*'yi düzenlediğiniz dosyayla değiştirin.
7. Qlik Sense Engine Service (QES) uygulamasını yeniden başlatın.

Qlik Sense hizmet olarak çalışıyorsa, bazı komutlar beklendiği gibi çalışmayabilir.

Örnek:

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

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

Field/Fields

Field ve **Fields** kod anahtar sözcükleri **Declare**, **Derive**, **Drop**, **Comment**, **Rename** ve **Tag/Untag** deyimlerinde kullanılır.

FlushLog

FlushLog deyimi, Qlik Sense uygulamasını kod belleğinin içeriğini kod günlük dosyasına yazmaya zorlar.

Söz Dizimi:

```
FlushLog
```

Arabelleğin içeriği günlük dosyasına yazılır. Bu komut, başarısız bir kod yürütmesinde kaybolabilecek verileri aldığınız için hata ayıklama amaçları için yararlı olabilir.

Örnek:

```
FlushLog;
```

Force

force deyimi, Qlik Sense uygulamasını alan değerlerini ve kendisinden sonra gelen **LOAD** ve **SELECT** deyimlerinin alan değerlerini yalnızca büyük harflerle, yalnızca küçük harflerle, her zaman ilk harfi büyük olarak veya göründüğü gibi (karışık) yorumlamaya zorlar. Bu deyim, tablodan alınan alan değerlerinin farklı kurallara göre ilişkilendirilmesini mümkün kılar.

Söz Dizimi:

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

Hiçbir şey belirtilmezse, büyük/küçük harf karışığını zorlama kabul edilir. force deyimi, yeni bir force deyimi yapılana dek geçerlidir.

Erişim bölümünde **force** deyiminin herhangi bir etkisi yoktur: Yüklenen tüm alan değerleri büyük/küçük harfe duyarlıdır.

Örnekler ve sonuçlar

Örnek	Sonuç

Bu örnekte ilk harflerin büyük olmasını zorlama gösterilmektedir.
 	
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Capitalization tablosu şu değerleri içerir:
	
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Tüm değerlerin ilk harfleri büyük yapılır.

Bu örnekte büyük harflere zorlama gösterilmektedir.
 	
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseUpper tablosu şu değerleri içerir:
	
	`AB`
	`CD`
	`EF`
	`GH`
	Tüm değerler büyük harftir.

Örnek	Sonuç

Bu örnekte küçük harflere zorlama gösterilmektedir.
 	
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseLower tablosu şu değerleri içerir:
	
	`ab`
	`cd`
	`ef`
	`gh`
	Tüm değerler küçük harftir.

Bu örnekte büyük/küçük harf karmasını zorlama gösterilmektedir.
 	
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseMixed tablosu şu değerleri içerir:
	
	`ab`
	`Cd`
	`eF`
	`GH`
	Tüm değerler kodda göründüğü gibidir.

Ayrıca bkz.

From

From kod anahtar sözcüğü, **Load** deyimlerinde bir dosyaya referansta bulunmak amacıyla ve **Select** deyimlerinde ise bir veritabanı tablosuna veya görünüme referansta bulunmak amacıyla kullanılır.

Load

LOAD deyimi, alanları bir dosyadan, kod içinde tanımlanmış verilerden, daha önceden yüklenmiş tablodan, web sayfasından, ardından gelen **SELECT** deyiminin sonucundan veya verileri otomatik olarak oluşturarak yükler. Analiz bağlantılarından da veri yüklenebilir.

Söz Dizimi:

```
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 ]
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
distinct	Yalnızca benzersiz kayıtları yüklemek istiyorsanız koşul olarak **distinct** öğesini kullanabilirsiniz. Çoğaltılmış kayıtlar varsa birinci örnek yüklenir. Önceki yüklemeleri kullanıyorsanız **distinct** yalnızca hedef tabloyu etkilediğinden birinci load deyimine **distinct** öğesini yerleştirmeniz gerekir.

Bağımsız Değişken	Açıklama
fieldlist	*fieldlist ::= (* | field {, * | field })* Yüklenecek alanların listesi. Alan listesi olarak * kullanılması tablodaki tüm alanları işaret eder. *field ::= (fieldref | expression) [***as** *aliasname]* Alan tanımı, her zaman için bir değişmez değeri, mevcut alana bir referansı veya bir ifadeyi içermelidir. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname*, tablodaki bir alan adıyla aynı olan metindir. Alan adının, örneğin boşluklar içeriyorsa, düz çift tırnak işaretleri veya köşeli ayraçlar içine alınması gerektiğini unutmayın. Alan adları kimi zaman açık şekilde kullanılabilir durumda olmayabilir. Bu durumda farklı gösterim kullanılır. *@fieldnumber*, sınırlanmış bir tablo dosyasındaki alan numarasını temsil eder. Önünde "@" yer alan pozitif bir tamsayı olmalıdır. Numaralandırma her zaman 1'den başlar ve alan sayısına kadar gider. *@startpos*:*endpos*, sabit uzunluklu kayıtların bulunduğu bir dosyada alanın başlangıç ve bitiş konumlarını temsil eder. Konumların her ikisi de pozitif tamsayı olmalıdır. Bu iki sayının öncesinde "@" gelmeli ve iki sayı iki nokta üst üste ile ayrılmalıdır. Numaralandırma her zaman 1'den başlar ve konumların sayısına kadar gider. Son alanda, bitiş konumu olarak **n** kullanılır. • *@startpos*:*endpos* öğesinin hemen ardından **I** veya **U** karakterleri gelirse, okunan baytlar imzalanmış ikili (**I**) veya imzalanmamış (**U**) tamsayı (Intel bayt sırası) olarak yorumlanır. Okunan konumların sayısı 1, 2 veya 4 olmalıdır. • *@startpos*:*endpos* öğesinin hemen ardından **R** karakteri gelirse, okunan baytlar ikili gerçek sayı (IEEE 32 bit ya da 64 bit kayan nokta) olarak yorumlanır. Okunan konumların sayısı 4 veya 8 olmalıdır. • *@startpos*:*endpos* öğesinin hemen ardından **B** karakteri gelirse, okunan baytlar COMP-3 standardına göre BCD (Binary Coded Decimal) sayıları olarak yorumlanır. İstenen sayıda bayt belirtilebilir. *expression*, aynı tablodaki bir veya birkaç alanı temel alan bir sayısal fonksiyon veya bir dize fonksiyonu olabilir. Daha fazla bilgi için ifadelerin söz dizimine bakın. Alana yeni bir ad atamak için **as** ifadesi kullanılır.

Bağımsız Değişken	Açıklama
from	Klasör veya web dosyası veri bağlantısı kullanılarak dosyadan veri yüklenmesi gerekiyorsa **from** kullanılır. *file ::= [path] filename* **Örnek:** ***'lib://Table Files/'*** Yol atlanırsa, Qlik Sense bu dosyayı **Directory** deyiminde belirtilen dizinde arar. **Directory** deyimi yoksa, Qlik Sense dosyayı *C:\Users\{user}\Documents\Qlik\Sense\Apps* çalışma dizininde arar. *Qlik Sense sunucu yüklemesinde, çalışma dizini Qlik Sense Repository Serviceiçinde belirtilir, varsayılan olarak C:\ProgramData\Qlik\Sense\Apps'tir.* *filename* standart DOS joker karakterlerini (* ve ?) içerebilir. Bu durum, belirtilen dizindeki tüm eşleşen dosyaların yüklenmesine neden olur. *format-spec ::= (fspec-item { , fspec-item })* Biçim belirtimi, ayraçlar içinde, birden fazla biçim belirtimi öğesinin listesinden oluşur. **Eski kod oluşturma modu** Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** ***c:\data*** • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** ***data*** • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** ***http://www.qlik.com***
from_field	Daha önceden yüklenmiş bir alandan veri yüklenmesi gerekirse **from_field** kullanılır. *fieldassource::=***(***tablename, fieldname***)*** Alan, daha önceden yüklenen *tablename* ve *fieldname* adıdır. *format-spec ::= (fspec-item {, fspec-item })* Biçim belirtimi, ayraçlar içinde, birden fazla biçim belirtimi öğesinin listesinden oluşur.

Bağımsız Değişken	Açıklama
inline	Verilerin kod içerisine yazılması ve dosyadan yüklenmemesi gerekirse **inline** kullanılır. *data ::= [text]* **inline** cümlesiyle girilen veriler çift tırnak işareti veya köşeli ayraçlar içine alınmalıdır. Bunlar arasındaki metinler bir dosyanın içeriğiyle aynı şekilde yorumlanır. Bu nedenle, bir metin dosyasında yeni satır eklerken, bunu aynı zamanda **inline** cümlesinin metninde de yapmalı, yani kodu yazarken Enter tuşuna basmalısınız. Sütun sayısı, ilk satıra bağlı olarak tanımlanır. *format-spec ::= (fspec-item {, fspec-item })* Biçim belirtimi, ayraçlar içinde, birden fazla biçim belirtimi öğesinin listesinden oluşur.
resident	Daha önceden yüklenmiş bir tablodan veri yüklenmesi gerekirse **resident** kullanılır. *table label*, asıl tabloyu oluşturan **LOAD** veya **SELECT** deyimlerinin önünde bulunan bir etikettir. Bu etiketin sonuna iki nokta üst üste eklenmelidir.
autogenerate	**autogenerate**, verilerin otomatik olarak Qlik Sense tarafından oluşturulması gerekiyorsa kullanılır. *size ::= number* *Number*, oluşturulacak kayıt sayını belirten bir tamsayıdır. Alan listesi, **Peek** fonksiyonuyla daha önce yüklenen bir tabloda tek bir alan değerine referansta bulunmadığınız sürece, harici veri kaynağından veya daha önce yüklenen tablodan veri gerektiren ifadeler içermemelidir.

Bağımsız Değişken	Açıklama
extension	Analiz bağlantılarından veri yükleyebilirsiniz. Sunucu tarafı uzantı (SSE) eklentisinde tanımlanan bir fonksiyonu çağırmak veya bir kodu değerlendirmek için **extension** cümlesini kullanmanız gerekir. SSE eklentisine tek bir tablo gönderebilirsiniz ve tek bir veri tablosu döndürülür. Eklenti, döndürülen alanların adlarını belirtmiyorsa alanlar, Field1, Field2 olarak adlandırılır ve bu şekilde devam eder. `Extension pluginname.functionname( tabledescription );` • SSE eklentisindeki bir fonksiyonu kullanarak veri yükleme *tabledescription ::= (table { ,tablefield})* Tablo alanları belirtmezseniz alanlar, yükleme sırasıyla kullanılır. • SSE eklentisindeki bir kodu değerlendirerek veri yükleme *tabledescription ::= (script, table { ,tablefield})* **Tablo alanı tanımında veri türü işleme** Veri türleri, analiz bağlantılarında otomatik olarak algılanır. Veriler bir sayısal değer ve en az bir NULL olmayan metin dizesi içermiyorsa alan, metin olarak değerlendirilir. Diğer tüm durumlarda sayısal olarak değerlendirilir. Alan adını bir **String()** veya **Mixed()** içine alarak veri türünü zorlayabilirsiniz. • **String()**, alanı metin olmaya zorlar. Alan sayısalsa, ikili değerin metin kısmı ayıklanır; dönüştürme gerçekleştirilmez. • **Mixed()**, alanı ikili olmaya zorlar. **String()** veya **Mixed()**, **extension** tablo alanı tanımları dışında kullanılamaz ve diğer Qlik Sense fonksiyonlarını bir tablo alanı tanımında kullanamazsınız. **Analiz bağlantıları hakkında daha fazla bilgi** Analiz bağlantılarını kullanabilmeniz için önce yapılandırmanız gerekir.
where	**where**, bir kaydın seçime dahil edilmesi gerekip gerekmediğini belirtmek için kullanılan bir cümledir. *criterion* değeri True ise seçim dahil edilir. *criterion*, mantıksal bir ifadedir.
while	**while**, bir kaydın tekrar tekrar okunması gerekip gerekmediğini belirtmek için kullanılan bir cümledir. *criterion* değeri True olduğu sürece aynı kayıt okunur. Kullanışlı olması için, **while** cümlesi genellikle **IterNo( )** fonksiyonunu içermelidir. *criterion*, mantıksal bir ifadedir.

Bağımsız Değişken	Açıklama
group by	**group by**, verilerin hangi alan üzerinde toplanması (gruplanması) gerektiğini tanımlamak için kullanılan bir cümledir. Toplama alanları yüklenen ifadelere bir şekilde dahil edilmelidir. Yüklenen ifadelerde toplama işlevleri dışında toplama alanlarından başka herhangi bir alan kullanılamaz. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by**, yerleşik tablonun kayıtlarını, **load** deyimi tarafından işlenmeden önce sıralamak için kullanılan bir cümledir. Yerleşik tablo bir veya daha fazla alana göre artan veya azalan olarak sıralanabilir. Sıralama, birincil olarak sayısal değere ve ikincil olarak da ulusal harmanlama düzenine göre yapılır. Bu cümle yalnızca veri kaynağı yerleşik bir tablo olduğunda kullanılabilir. Düzenleme alanları, yerleşik tablonun hangi alana göre sıralanacağını belirler. Bu alan, adıyla veya yerleşik tablodaki sayısıyla belirlenebilir (birinci alan 1 numaradır). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder*, artan için *asc* veya azalan için *desc* şeklindedir. Herhangi bir *sortorder* belirtilmezse *asc* olduğu varsayılır. *fieldname*, *path*, *filename* ve *aliasname* sırasıyla kendi adlarının ifade ettiklerini temsil eden metin dizeleridir. Kaynak tablodaki herhangi bir alan *fieldname* olarak kullanılabilir. Ancak, as cümlesiyle oluşturulan alanlar (*aliasname*) kapsam dışıdır ve aynı **load** deyiminin içerisinde kullanılamaz.

Bir **from**, **inline**, **resident**, **from_field, extension** veya **autogenerate** cümlesi aracılığıyla herhangi bir veri kaynağı verilmezse, veriler hemen arkadan gelen **SELECT** veya **LOAD** deyiminin sonucundan yüklenir. Bunun ardından gelen deyimin bir öneki bulunmamalıdır.

Örnekler:

Farklı dosya biçimlerini yükleme
Varsayılan seçeneklerle bir sınırlanmış veri dosyası yükleyin:

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

Kütüphane bağlantısından sınırlı veri dosyası yükleyin (DataFiles):

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

Kütüphane bağlantısından tüm sınırlı veri dosyalarını yükleyin (DataFiles):

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

Sınırlayıcı olarak virgül belirterek ve eklenmiş etiketlerle bir sınırlanmış dosya yükleyin:

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

Sınırlayıcı olarak sekme belirterek ve eklenmiş etiketlerle bir sınırlanmış dosya yükleyin:

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

Eklenmiş üst bilgilerle bir dif dosyası yükleyin:

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

Üst bilgileri olmayan sabit bir kayıt dosyasından üç alan yükleyin:

```
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);
```

Mutlak yol belirterek bir QVX dosyası yükleyin:

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

Web dosyalarını yükleme

Web dosyası veri bağlantısında ayarlanan varsayılan URL'den yükleme:

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

Belirli bir URL'den yükleme ve web dosyası veri bağlantısında ayarlanan URL'yi geçersiz kılma:

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

Dolar işareti genişletmesini kullanarak bir değişkende ayarlanan belirli bir URL'den yükleme:

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

Belirli alanları seçme, alanları yeniden adlandırma ve hesaplama

Sınırlanmış dosyadan yalnızca üç belirli alanı yükleyin:

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

Etiketleri olmayan bir dosyayı yüklerken ilk alanı A ve ikinci alanı B olarak yeniden adlandırın:

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

FirstName, bir boşluk karakteri ve LastName birleşimi olarak Name öğesini yükleyin:

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

Quantity, Price ve Value (Quantity ve Price öğelerinin çarpımı) öğelerini yükleyin:

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

Belirli kayıtları seçme

Yalnızca benzersiz kayıtları yükleyin; çoğaltılan kayıtlar atılır:

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

Yalnızca Litres alanının sıfır üzerinde bir değere sahip olduğu kayıtları yükleyin:

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

Dosyada olmayan ve otomatik olarak oluşturulan verileri yükleme
CatID ve Category adında iki alan olmak üzere satır içi verileri içeren bir tablo yükleyin:

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

UserID, Password ve Access adında üç alan olmak üzere satır içi verileri içeren bir tablo yükleyin:

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

10.000 satırlı bir tablo yükleyin. A alanı okunan kayıt sayısını (1,2,3,4,5...) ve B alanı da 0 ile 1 arasında rastgele bir sayı içerecektir:

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

autogenerate deyiminden sonra paranteze izin verilir, ancak bu gerekli değildir.

Daha önce yüklenmiş bir tablodan verileri yükleme
İlk olarak bir sınırlanmış tablo dosyası yüklüyor ve tab1 olarak adlandırıyoruz:

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

Önceden yüklenmiş tab1 tablosundan dosyaları tab2 olarak yükleyin:

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

Önceden yüklenmiş tab1 tablosundan dosyaları yükleyin; ancak yalnızca A değerinin B değerinden büyük olduğu kayıtları yükleyin:

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

Önceden yüklenmiş tab1 tablosundan alanları, A ölçütüne göre sıralanmış olarak yükleyin:

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

Önceden yüklenmiş tab1 tablosundan alanları, birinci alana ve sonra da ikinci alana göre sıralanmış olarak yükleyin:

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

Önceden yüklenmiş tab1 tablosundan değerleri, C ölçütüne göre azalan düzende, ardından B ölçütüne göre artan düzende ve sonra da ilk alana göre azalan düzende sıralanmış olarak yükleyin:

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

Daha önce yüklenmiş alanlardan verileri yükleme
Daha önce yüklenmiş Characters tablosundan Types alanını A olarak yükleyin:

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

Ardından gelen tablodan verileri yükleme (öncelikli yükleme)
Ardından gelen **SELECT** deyiminde yüklenen Table1 öğesinden A, B ve hesaplanan X ve Y alanlarını yükleyin:

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

Verileri gruplandırma
ArtNo öğesine göre gruplandırılmış (toplanmış) alanları yükleyin:

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

Week ve ArtNo öğesine göre gruplandırılmış (toplanmış) alanları yükleyin:

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

Bir kaydı tekrar tekrar okuma
Bu örnekte, her bir öğrencinin notlarını tek bir alana sıkıştırılmış olarak içeren Grades.csv adında bir giriş dosyamız var:

```
Student,Grades
Mike,5234
John,3345
Pete,1234
Paul,3352
```

Notlar, 1-5 ölçeğinde Math, English, Science ve History derslerini temsil etmektedir. **IterNo( )** fonksiyonunun sayaç olarak kullanıldığı bir **while** cümlesi ile her bir kaydı birkaç kez okuyarak, notları ayrı değerler halinde ayırabiliriz. Her okumada, öğrenci notu **Mid** fonksiyonu ile ayrıştırılıp Grade alanına depolanır ve ders de **pick** fonksiyonunun kullanımıyla seçilip Subject alanına depolanır. Son **while** cümlesi, tüm notların (bu örnekte öğrenci başına dört not) okunduğunu kontrol etmek için kullanılan ve okunduysa, bir sonraki öğrenci kaydının okunması gerektiği anlamına gelen sınamayı içerir.

```
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));
```

Sonuçta şu verileri içeren bir tablo ortaya çıkar:

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

Analiz bağlantılarından yükleme
Aşağıdaki örnek veriler kullanılır.

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

Bir fonksiyon kullanarak veri yükleme

Bu örneklerde, *Calculate(Parameter1, Parameter2)* özel fonksiyonunu içeren *P* adlı bir analiz bağlantısı eklentimizin olduğunu varsayarız. Fonksiyon, *Field1* ve *Field2* alanlarını içeren *Results* tablosunu döndürür.

```
Load * Extension P.Calculate( Values{A, C} );
```
A ve C alanları fonksiyona gönderilirken döndürülen tüm alanları yükleyin.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
A ve C alanları fonksiyona gönderilirken yalnızca Field1 alanını yükleyin.

```
Load * Extension P.Calculate( Values );
```
A ve B alanları fonksiyona gönderilirken döndürülen tüm alanları yükleyin. Alanlar belirtilmediğinden, tabloda birinci olarak sıralanan A ve B kullanılır.

```
Load * Extension P.Calculate( Values {C, C});
```
C alanı, fonksiyonun her iki parametresine gönderilirken döndürülen tüm alanları yükleyin.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Dize olarak zorlanan A alanı ve sayısal olarak zorlanan B alanı fonksiyona gönderilirken döndürülen tüm alanları yükleyin.

Bir kodu değerlendirerek veri yükleme

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
A ve B değerlerini gönderirken q kodu tarafından döndürülen tabloyu yükleyin.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
A ve B değerleri gönderilirken My_R_Script değişkeninde saklanan kod tarafından döndürülen tabloyu yükleyin.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
B değerleri, D, A ve C olarak yeniden adlandırılmış şekilde gönderilirken My_R_Script değişkeninde saklanan kod tarafından döndürülen tabloyu yükleyin. * işareti kullanıldığında, referansta bulunulmayan diğer kalan alanlar gönderilir.

DataFiles bağlantılarının dosya uzantısı büyük-küçük harfe duyarlıdır. Örneğin: .qvd.

Biçimlendirme belirtim öğeleri

Her bir biçimlendirme belirtim öğesi tablo dosyasının belirli bir özelliğini tanımlar.

`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`]

Karakter kümesi

Karakter kümesi, dosyada kullanılan karakter kümesini tanımlayan, **LOAD** deyimine yönelik bir dosya tanımlayıcısıdır.

ansi, **oem** ve **mac** tanımlayıcıları, QlikView uygulamasında kullanılmıştır ve çalışmaya devam etmektedir. Ancak, Qlik Sense ile **LOAD** deyimi oluşturulurken bunlar oluşturulmaz.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
utf8	UTF-8 karakter kümesi
unicode	Unicode karakter kümesi
ansi	Windows, kod sayfası 1252
oem	DOS, OS/2, AS400 ve diğerleri

Bağımsız Değişken	Açıklama
mac	Kod sayfası 10000
codepage is	**codepage** belirticisi ile herhangi bir Windows kod sayfasını *N* olarak kullanmak mümkündür.

Sınırlamalar:

oem karakter kümesinden dönüşüm MacOS için uygulanmaz. Hiçbir şey belirtilmezse, Windows altında kod sayfası 1252 varsayılır.

Örnek:

```
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)
```

Ayrıca bkz.

Load (page 98)

Tablo biçimi

Tablo biçimi, dosya türünü tanımlayan **LOAD** deyimi için bir dosya belirticisidir. Hiçbir şey belirlenmezse, dosyanın bir *.txt* dosyası olduğu kabul edilir.

Tablo biçimi türleri

Tür	Açıklama
txt	Sınırlanmış metin dosyasında, tablodaki sütunlar sınırlayıcı bir karakter ile ayrılır.
fix	Sabit kayıt dosyasında, her alan tam olarak belirli bir sayıda karakterden oluşur. Tipik olarak, birçok sabit kayıt uzunluğu dosyası satır besleme ile ayrılmış kayıtlar içerir, ancak kayıt boyutunu bayt cinsinden belirtmek veya **Record is** ile birden fazla satıra yaymak için daha gelişmiş seçenekler vardır. *Veriler çok baytlı karakterler içeriyorsa, biçimin bayt cinsinden sabit bir uzunluğu temel alması nedeniyle alan sonlarının hizası bozulabilir.*
dif	*.dif* dosyasında (Data Interchange Format), kullanılan tabloyu tanımlamaya yönelik özel bir biçim.
biff	Qlik Sense ayrıca, standart Excel dosyalarındaki verileri *biff* biçiminin (Binary Interchange File Format) yardımıyla yorumlayabilir.
ooxml	Excel 2007 ve sonraki sürümler ooxml *.xslx* biçimini kullanır.

Tür	Açıklama
html	Tablo bir html sayfasının veya dosyasının parçasıysa html kullanılmalıdır.
xml	xml (Extensible Markup Language), metin biçiminde veri yapılarını temsil etmek için kullanılan bir ortak biçimlendirme dilidir.
qvd	*qvd* biçimi, bir Qlik Sense uygulamasından dışa aktarılan özel QVD dosyaları biçimidir.
qvx	*qvx*, Qlik Sense uygulamasına yüksek performanslı çıkış sağlayan dosya/klasör biçimidir.

Delimiter is

Ayrılmış tablo dosyaları için, **delimiter is** belirticisi aracılığıyla rastgele bir ayırıcı belirtilebilir. Bu belirtici, yalnızca ayrılmış .txt dosyaları için geçerlidir.

Söz Dizimi:

```
delimiter is char
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
char	127 ASCII karakterinden tek bir karakter belirtir.

Ek olarak aşağıdaki değerler kullanılabilir:

İsteğe bağlı değerler

Değer	Açıklama
'\t'	Tırnak işaretleriyle veya tırnak işaretleri olmadan, bir sekme işaretini temsil eder.
'\\'	Ters eğik çizgi (\ ) karakterini temsil eder.
'spaces'	Bir veya birden fazla boşluğun tüm bileşimlerini temsil eder. CR ve LF haricinde, 32'nin altındaki bir ASCII değerine sahip yazdırılamayan karakterler boşluk olarak yorumlanır.

Hiçbir şey belirtilmezse **delimiter is ','** olduğu varsayılır.

Örnek:

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

Ayrıca bkz.

Load (page 98)

No eof

no eof belirticisi, ayrılmış **.txt** dosyalarını yüklerken dosya sonu karakterini göz ardı etmek için kullanılır.

Söz Dizimi:

```
no eof
```

no eof belirticisi kullanılırsa, aksi durumda dosya sonunu belirten 26 kod noktalı karakterler göz ardı edilir ve bir alan değerinin parçası olabilirler.

Bu yalnızca sınırlanmış metin dosyaları için geçerlidir.

Örnek:

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

Ayrıca bkz.

Load (page 98)

Labels

Labels, dosya içerisinde alan adlarının nerede bulunabileceğini tanımlayan **LOAD** deyimi için dosya belirticisidir.

Söz Dizimi:

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

Alan adları dosyanın farklı yerlerinde bulunabilir. İlk kayıt alan adlarını içeriyorsa **embedded labels** kullanılmalıdır. Bulunabilecek herhangi bir alan adı yoksa **no labels** kullanılmalıdır. *dif* dosyalarında bazen açık alan adlarına sahip ayrı bir üst bilgi bölümü kullanılır. Böyle bir durumda **explicit labels** kullanılmalıdır. Hiçbir şey belirtilmezse, *dif* dosyaları için de **embedded labels** kabul edilir.

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)
```

Ayrıca bkz.

Load (page 98)

Header is

Tablo dosyalarındaki üst bilgi boyutunu belirler. Rastgele üst bilgi uzunluğu **header is** tanımlayıcısıyla belirlenebilir. Üst bilgi, Qlik Sense tarafından kullanılmayan metin bölümüdür.

Söz Dizimi:

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

Üst bilgi uzunluğu bayt (**header is n**) veya satır (**header is line** ya da **header is n lines**) cinsinden verilebilir. **n**, üst bilgi uzunluğunu temsil eden, pozitif bir tamsayı olmalıdır. Belirtilmediği takdirde **header is 0** olduğu varsayılır. **header is** belirticisi yalnızca tablo dosyalarıyla ilgilidir.

Örnek:

Bu, Qlik Sense tarafından veri olarak yorumlanmaması gereken üst bilgi metin satırı içeren bir veri kaynağı tablosu örneğidir.

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

header is 1 lines belirticisi kullanıldığında ilk satır veri olarak yüklenmez. Örnekte, **embedded labels** belirticisi Qlik Sense uygulamasına, ilk hariç tutulmayan satırı alan etiketleri içeriyormuş gibi yorumlamasını söyler.

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

Sonuç, Col1 ve Col2 alanlarına sahip iki alanlı bir tablodur.

Ayrıca bkz.

Load (page 98)

Record is

Sabit kayıt uzunluğu dosyaları için, kayıt uzunluğu **record is** belirticisiyle belirtilmelidir.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n	Bayt cinsinden kayıt uzunluğunu belirtir.
line	Tek bir satır olarak kayıt uzunluğunu belirtir.
n lines	Satır olarak kayıt uzunluğunu belirtir; burada n kayıt uzunluğunu temsil eden bir pozitif tamsayıdır.

Sınırlamalar:

record is belirticisi yalnızca **fix** dosyalarıyla ilgilidir.

Ayrıca bkz.

Load (page 98)

Quotes

Quotes, tırnak işaretlerinin kullanılıp kullanılamayacağını ve tırnak işaretleri ile ayırıcılar arasındaki önceliği tanımlayan, **LOAD** deyimine yönelik bir dosya tanımlayıcısıdır. Yalnızca metin dosyalarına yöneliktir.

Söz Dizimi:

```
no quotes
msq
```

Belirtici atlandığı takdirde standart tırnak işareti uygulaması kullanılır; yani " " veya ' ' kullanılabilir. Ancak bu yalnızca bunların bir alan değerinin ilk ve son boş olmayan karakteri olmaları durumunda geçerlidir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
no quotes	Bir metin dosyasında tırnak işaretleri kabul edilmeyecek olduğunda kullanılır.
msq	Alanlarda birden çok satırlı içeriğe olanak tanıyan modern tırnak işareti uygulaması stilini belirtmek için kullanılır. Satır sonu karakterleri içeren alanlar çift tırnak içine alınmalıdır. msq seçeneğine yönelik bir sınırlama, alan içeriğinde ilk veya son karakter olarak görünen bir adet çift tırnak (") karakterinin birden çok satırlı içeriğin başlangıcı veya sonu olarak yorumlanacak olmasıdır ve bu da yüklenen veri kümesinde öngörülemeyen sonuçlara neden olabilir. Bu durumda, belirticiyi atarak bunun yerine standart tırnak uygulamasını kullanmanız gerekir.

XML

Bu kod belirticisi xml dosyalarını yüklerken kullanılır. **XML** belirticisi için geçerli seçenekler söz diziminde listelenir.

Qlik Sense uygulamasında DTD dosyaları yükleyemezsiniz.

Söz Dizimi:

```
xmlsimple
```

Ayrıca bkz.

Load (page 98)

KML

Harita görselleştirmesinde kullanılacak KML dosyaları yüklenirken kod belirtici kullanılır.

Söz Dizimi:

```
kml
```

KML dosyası poligonlarla gösterilen alan verilerini (örneğin, ülkeler veya bölgeler), satır verilerini (örneğin, yollar) ya da [enl, boy] biçiminde noktalarla gösterilen nokta verilerini (örneğin, şehirler veya yerler) temsil edebilir.

URL is

Bu kod belirticisi, bir web dosyası yüklenirken web dosyası veri bağlantısının URL'sini ayarlamak için kullanılır.

Söz Dizimi:

```
URL is string
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	Yüklenecek dosyanın URL'sini belirtir. Bu, kullanılan web dosyası bağlantısında ayarlanan URL'yi geçersiz kılar.

Sınırlamalar:

URL is belirticisi yalnızca web dosyalarıyla ilgilidir. Mevcut bir web dosyası veri bağlantısı kullanmanız gerekir.

Ayrıca bkz.

Load (page 98)

userAgent is

Bu kod belirticisi, web dosyası yüklenirken tarayıcı kullanıcı aracısını ayarlamak için kullanılır.

Söz Dizimi:

```
userAgent is string
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	Tarayıcı kullanıcı aracısı dizesini belirtir. Bu, varsayılan "Mozilla/5.0" tarayıcı kullanıcı aracısını geçersiz kılar.

Sınırlamalar:

userAgent is belirticisi yalnızca web dosyalarıyla ilgilidir.

Ayrıca bkz.

Load (page 98)

Let

let deyimi **set** deyiminin tamamlayıcısıdır ve kod değişkenlerini tanımlamak için kullanılır. **let** deyimi, **set** deyiminin aksine "=" işaretinin sağındaki ifadeyi, kodun çalışma zamanında değişkene atanmadan önce değerlendirir.

Söz Dizimi:

```
Let variablename=expression
```

Örnekler ve sonuçlar:

Örnek	Sonuç

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` öğesi ' 3+4 ' olarak değerlendirilir
	
	`$(y)` öğesi ' 7 ' olarak değerlendirilir
	
	`$(z)` öğesi ' 8 ' olarak değerlendirilir
	
	Set ve **Let** deyimi arasındaki farka dikkat edin. **Set** deyimi değişkene "3+4" dizesini atar, **Let** deyimi ise dizeyi değerlendirir ve değişkene 7 değerini atar.

Let T=now( );	`$(T)` öğesine geçerli zamanın değeri verilir.

Loosen Table

Bir veya daha fazla Qlik Sense dahili veri tablosu, **Loosen Table** deyimi kullanılarak kod yürütmesi sırasında açık şekilde gevşek bağlı olarak bildirilebilir. Bir tablo gevşek bağlı olduğunda, tabloda bulunan alan değerleri arasındaki tüm ilişkiler kaldırılır. Benzer bir etki, gevşek bağlı tablonun her bir alanının bağımsız, ilişkisiz tablolar olarak yüklenmesiyle elde edilebilir. Gevşek bağlı özelliği, test sırasında veri yapısının farklı bölümlerinin geçici olarak ayrı tutulmasında yararlı olabilir. Gevşek bağlı bir tablo, tablo görüntüleyicisinde noktalı çizgilerle gösterilebilir. Kod içerisinde bir veya daha fazla **Loosen Table** deyimi kullanılması, Qlik Sense uygulamasının kodu yürütmeden önce gevşek bağlı tablolarda yapılan ayarları göz ardı etmesine neden olur.

Söz Dizimi:

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

Loosen Table ve **Loosen Tables** sözdizimlerinden herhangi biri kullanılabilir.

> *Qlik Sense uygulamasının, veri yapısında, etkileşimli olarak veya kod içinde açıkça gevşek bağlı olduğu bildirilen tablolar ile bölünemeyen döngüsel referanslar bulması durumunda, herhangi bir döngüsel referans kalmayana kadar bir veya daha fazla ek tablo gevşek bağlı olmaya zorlanır. Bu durum gerçekleştiğinde,* ***Döngü Uyarısı*** *diyalog penceresi uyarı verir.*

Örnek:

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

Map

map ... using deyimi, belirli bir alan değerini veya ifadesini belirli bir eşleme tablosunun değerlerine eşlemek için kullanılır. Eşleme tablosu **Mapping** deyimi aracılığıyla oluşturulur.

Söz Dizimi:

```
Map fieldlist Using  mapname
```

Otomatik eşleme, **Map … Using** deyimi sonrasında yüklenen alanlar için kodun sonuna dek veya bir **Unmap** deyimiyle karşılaşıncaya dek yapılır.

Eşleme işlemi, alanın Qlik Sense içindeki dahili tabloda saklanmasıyla sonuçlanacak olaylar zincirinde son aşama olarak gerçekleştirilir. Bu da eşlemenin bir ifadenin parçası olarak bir alan adıyla her karşılaşıldığına değil; ancak değer dahili tabloda alan adı altında saklandığında gerçekleştirileceği anlamına gelir. İfade seviyesinde eşleme gerekliyse, bunun yerine **Applymap()** fonksiyonu kullanılmalıdır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fieldlist	Kod içinde bu noktadan eşlenmesi gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.
mapname	Bir **mapping load** veya **mapping select** deyiminde daha önce okunmuş bir eşleme tablosunun adı.

Örnekler ve sonuçlar:

Örnek	Sonuç
Map Country Using Cmap;	Country alanının, Cmap eşlemesi kullanılarak eşlenmesini sağlar.
Map A, B, C Using X;	A, B ve C alanlarının, X eşlemesi kullanılarak eşlenmesini sağlar.
Map * Using GenMap;	Tüm alanların GenMap kullanılarak eşlenmesini sağlar.

NullAsNull

NullAsNull deyimi, NULL değerlerin **NullAsValue** deyimi tarafından daha önce ayarlanmış dize değerlerine dönüştürülmesi işlemini kapatır.

Söz Dizimi:

```
NullAsNull *fieldlist
```

NullAsValue deyimi bir anahtar olarak çalışır ve **NullAsValue** veya **NullAsNull** deyimi kullanılarak kod içinde birden fazla kez açılabilir ve kapatılabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	**NullAsNull** deyiminin açılması gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.

Örnek:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

NullAsValue deyimi, hangi alanlar için NULL öğesinin bir değere döndürülmesi gerektiğini belirtir.

Söz Dizimi:

```
NullAsValue *fieldlist
```

Varsayılan olarak, Qlik Sense, NULL değerleri eksik veya tanımlanmamış varlıklar olarak dikkate alır. Bununla birlikte, belirli bazı veritabanı bağlamları NULL değerlerin basit bir eksik değerden çok özel değerler olarak dikkate alınması gerektiğini belirtir. NULL değerlerin normal olarak diğer NULL değerlerle bağlanmasına izin verilmemesi durumu, **NullAsValue** deyimi aracılığıyla askıya alınabilir.

NullAsValue deyimi bir anahtar olarak çalışır ve takip eden yükleme deyimlerinde işler. Bu deyim, **NullAsNull** deyimi aracılığıyla tekrar kapatılabilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	**NullAsValue** deyiminin açılması gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.

Örnek:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Qualify deyimi, alan adlarının nitelendirilmesi, yani alan adlarının tablo adıyla aynı öneki alması özelliğini açmak için kullanılır.

Söz Dizimi:

```
Qualify *fieldlist
```

Farklı tablolardaki aynı adlı alanlar arasında otomatik birleştirme, alan adını kendisine ait tablo adıyla niteleyen **qualify** deyimi aracılığıyla askıya alınabilir. Koşullara uyduğu takdirde, alan adları bir tabloda bulunduğunda yeniden adlandırılır. Yeni ad *tablename.fieldname* biçiminde olur. *Tablename*, geçerli tablonun etiketine eşdeğerdir veya bir etiket yoksa **LOAD** ve **SELECT** deyimlerindeki **from** öğesinden sonra görünen ada eşdeğerdir.

Niteleme, **qualify** deyiminden sonra yüklenen tüm alanlar için yapılır.

Niteleme, varsayılan olarak, kod yürütmesinin başında her zaman kapalıdır. Bir alan adının nitelenmesi, istenildiği zaman **qualify** deyimi kullanılarak etkinleştirilebilir. Niteleme, istenildiği zaman **Unqualify** deyimi kullanılarak kapatılabilir.

> ***qualify*** *deyimi kısmi yeniden yüklemeyle birlikte kullanılmamalıdır.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	Nitelemenin açılması gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.

Example 1:

```
Qualify B;
LOAD A,B from x.csv;
LOAD A,B from y.csv;
```

İki tablo (**x.csv** ve **y.csv**) yalnızca **A** aracılığıyla ilişkilidir. Sonuçta ortaya üç alan çıkar: A, x.B, y.B.

Example 2:

Alışık olunmayan bir veritabanında, aşağıdaki örnekte gösterildiği gibi, yalnızca bir veya birkaç alanın ilişkilendirildiğinden emin olarak başlamak çoğunlukla faydalı olur:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
```

```
SQL SELECT * from tab3;
```
tab1, *tab2* ve *tab3* tabloları arasındaki ilişkilendirmeler için yalnızca **TransID** alanı kullanılır.

Rem

rem deyimi, koda açıklama veya yorum eklemek veya kod deyimlerini kaldırmadan geçici olarak etkinliklerini kaldırmak için kullanılır.

Söz Dizimi:

```
Rem string
```

rem ile sonraki noktalı virgül (**;**) arasındaki tüm içerik yorum olarak ele alınır.

Kodda yorum yapmak için iki alternatif yöntem kullanılabilir:

1. İlgili bölümü **/*** ve ***/** arasına alarak, iki tırnak işaretinin arasında olmamak kaydıyla, kod içinde herhangi bir konumda yorum oluşturulabilir.
2. Kodda **//** yazıldığında, aynı satır üzerinde sağa doğru devam eden tüm metin yorum haline gelir. (Bir İnternet adresinin parçası olarak kullanılmış olabilecek //: özel durumu unutulmamalıdır.)

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	Rastgele seçilen bir metin.

Örnek:

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

Rename

Rename kod anahtar sözcüğü, zaten yüklenmiş tabloları veya alanları yeniden adlandırmak için kullanılabilir.

Rename field

Bu kod fonksiyonu, bir veya daha fazla var olan Qlik Sense alanını yüklendikten sonra yeniden adlandırır.

Qlik Sense içindeki bir alanda veya fonksiyonda bir değişkene aynı adı vermek önerilmez.

rename field ve **rename fields** sözdizimlerinden herhangi biri kullanılabilir.

Söz Dizimi:

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

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

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
mapname	Bir veya daha fazla eski ve yeni alan adı çifti içeren önceden yüklenmiş eşleme tablosunun adı.
oldname	Eski dosya adı.
newname	Yeni dosya adı.

Sınırlamalar:

İki alanın adını aynı olacak şekilde değiştiremezsiniz.

Example 1:

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

Example 2:

```
FieldMap:
Mapping SQL SELECT oldnames, newnames from datadictionary;
Rename Fields using FieldMap;
```

Rename table

Bu kod fonksiyonu, bir veya daha fazla var olan Qlik Sense dahili tabloyu yüklendikten sonra yeniden adlandırır.

rename table ve **rename tables** sözdizimlerinden herhangi biri kullanılabilir.

Söz Dizimi:

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

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
mapname	Bir veya daha fazla eski ve yeni tablo adı çifti içeren önceden yüklenmiş eşleme tablosunun adı.
oldname	Eski tablo adı.
newname	Yeni tablo adı.

Sınırlamalar:

Farklı şekilde adlandırılmış iki tablo, aynı ada sahip olacak şekilde yeniden adlandırılamaz. Kod, tablonun adını mevcut bir tablonun adıyla değiştirmeye çalışırsanız hata oluşturacaktır.

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

Akıllı aramada alanları dahil etmek veya hariç tutmak için **Search** deyimi kullanılır.

Söz Dizimi:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Dahil edilecek alanlarla ilgili seçiminizi daraltmak için çeşitli Search deyimleri kullanabilirsiniz. Deyimler üstten alta doğru değerlendirilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	Akıllı aramada aramalara dahil edilecek veya aramalardan hariç tutulacak alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.

Örnek:

Arama örnekleri

Deyim	Açıklama
`Search Include *;`	Akıllı aramadaki aramalara tüm alanları dahil edin.

Deyim	Açıklama
`Search Exclude [*ID];`	ID ile biten tüm alanları akıllı aramadaki aramalardan hariç tutun.
`Search Exclude '*ID';`	ID ile biten tüm alanları akıllı aramadaki aramalardan hariç tutun.
`Search Include ProductID;`	Akıllı aramadaki aramalara ProductID alanını dahil edin.

Bu üç deyimin birleştirilmiş sonucu (bu sırayla), ProductID alanı dışında ID ile biten tüm alanların akıllı aramadaki aramalardan hariç tutulmasıdır.

Section

section deyimiyle, sonraki **LOAD** ve **SELECT** deyimlerinin veri veya erişim haklarının bir tanımı olarak ele alınmasına ilişkin seçimi tanımlamak mümkündür.

Söz Dizimi:

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

Hiçbir şey belirtilmezse **section application** olduğu varsayılır. **section** tanımı, yeni bir **section** deyimi belirtilene kadar geçerlidir.

Örnek:

```
Section access;
Section application;
```

Select

Bir ODBC veri kaynağından veya bir OLE DB sağlayıcısından alanların seçilmesi, standart SQL **SELECT** deyimleriyle gerçekleştirilir. Bununla birlikte, **SELECT** deyimlerinin kabul edilip edilmemesi, kullanılan ODBC sürücüsüne veya OLE DB sağlayıcısına bağlıdır. **SELECT** ifadesinin kullanımı kaynağa yönelik açık bir veri bağlantısı gerektirir.

Söz Dizimi:

```
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 ]
```

Ayrıca, bazen birkaç **SELECT** deyimi **union** işlecinin kullanımıyla tek bir deyimde birleştirilebilir:

```
selectstatement Union selectstatement
```

SELECT deyimi ODBC sürücüsü veya OLE DB sağlayıcısı tarafından yorumlandığından, ODBC sürücülerinin veya OLE DB sağlayıcısının özelliklerine bağlı olarak genel SQL söz diziminden sapmalar olabilir. Örneğin:

- **as** öğesine bazen izin verilmez, yani *aliasname* öğesinin *fieldname* öğesinden hemen sonra gelmesi gerekir.
- *aliasname* kullanılırsa **as** bazen zorunlu olur.
- **distinct**, **as**, **where**, **group by**, **order by** veya **union** bazı durumlarda desteklenmez.
- ODBC sürücüsü bazen yukarıda listelenen tüm farklı tırnak işaretlerini kabul etmez.

*Bu, SQL **SELECT** deyimi hakkında eksiksiz bir açıklama değildir! Örneğin, **SELECT** deyimleri iç içe geçirilebilir, tek bir **SELECT** deyiminde birkaç birleştirme yapılabilir, ifadelerde izin verilen fonksiyonların sayısı bazen çok fazla olabilir vs.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
distinct	**distinct**, seçilen alanlardaki değerlerin çoğaltılmış bileşimlerinin yalnızca bir kez yüklenmesi gerektiğinde kullanılan bir koşuldur.
distinctrow	**distinctrow**, kaynak tablodaki çoğaltılmış kayıtların yalnızca bir kez yüklenmesi gerektiğinde kullanılan bir koşuldur.
fieldlist	**`fieldlist ::= (*| field ) {, field }`** Seçilecek alanların listesi. Alan listesi olarak * kullanılması tablodaki tüm alanları işaret eder. **`fieldlist ::= field {, field }`** Virgülle ayrılmış şekilde, bir veya daha fazla alanı içeren liste. **`field ::= ( fieldref| expression ) [as aliasname ]`** İfade örneğin diğer bir veya birkaç alanı temel alan bir sayısal fonksiyon veya dize fonksiyonu olabilir. Genellikle kabul edilen işleç ve fonksiyonların bazıları şunlardır: +, -, *, /, & (dize birleşimi), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname) vb. Daha fazla bilgi için ODBC sürücüsünün belgelerine bakın. **`fieldref ::= [ tablename. ] fieldname`** **tablename** ve **fieldname** belirttikleri içerikle aynı olan metin dizeleridir. Örneğin, boşluk içermeleri durumunda düz çift tırnak işaretleri içine alınmaları gerekir. **as** cümlesi alana yeni bir ad atamak için kullanılır.

Bağımsız Değişken	Açıklama
from	**tablelist ::= table {, table }** Alanların seçileceği tabloların listesi. **table ::= tablename [[as] aliasname]** **tablename** tırnak içine alınabilir veya alınmayabilir.
where	**where**, bir kaydın seçime dahil edilmesi gerekip gerekmediğini belirtmek için kullanılan bir cümledir. **criterion**, bazen oldukça karmaşık olabilen bir mantıksal ifadedir. Kabul edilen işleçlerden bazıları şunlardır: Sayısal işleçler ve fonksiyonlar, **=**, **<>** veya **#(**eşit değildir), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** ve ayrıca yeni **SELECT** deyimleri. Daha fazla bilgi için ODBC sürücüsü veya OLE DB sağlayıcısı ile ilgili belgelere bakın.
group by	**group by**, birkaç kaydı bir kayıtta toplamak (gruplandırmak) için kullanılan bir cümledir. Bir grup içinde, belirli bir alan için tüm kayıtlar aynı değere sahip olmalıdır; aksi takdirde alan yalnızca bir ifadenin içinden (örneğin, toplam veya ortalama olarak) kullanılabilir. Bir veya birkaç alanı temel alan ifade, alan sembolünün ifadesinde tanımlanmıştır.
having	**having** öğesi, **where** cümlesinin kayıtları nitelemek için kullanılmasına benzer şekilde grupları nitelemek için kullanılan bir cümledir.
order by	**order by** öğesi, **SELECT** deyiminin sonuç olarak elde edilen tablosunun sıralama düzenini belirtmek için kullanılan bir cümledir.
join	**join**, birkaç tablonun bir tabloda birleştirilip birleştirilmeyeceğini belirten bir niteleyicidir. Alan adları ve tablo adları, boşluk veya ulusal karakter kümelerine ait harfler içermeleri durumunda tırnak içine alınmalıdır. Kod Qlik Sense tarafından otomatik olarak oluşturulduğunda, burada kullanılan tırnak işareti **Connect** deyimindeki veri kaynağının veri kaynağı tanımında belirtilen ODBC sürücüsü veya OLE DB sağlayıcısı tarafından tercih edilen tırnak işaretidir.

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

set deyimi kod değişkenlerini tanımlamak için kullanılır. Bunlar dizelerin, yolların, sürücülerin ve benzeri öğelerin yerini alması için kullanılabilir.

Söz Dizimi:

```
Set variablename=string
```

Example 1:

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

Example 2:

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

Example 3:

```
Set BudgetYear=2012;
```

Sleep

sleep deyimi kod yürütmesini belirtilen süre kadar duraklatır.

Söz Dizimi:

```
Sleep n
```

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
n	Milisaniye cinsinden belirtilir; burada *n* bir pozitif tamsayıdır ve 3600000 (yani, 1 saat) değerinden büyük olamaz. Değer bir ifade olabilir.

Example 1:

```
Sleep 10000;
```

Example 2:

```
sleep t*1000;
```

SQL

SQL deyimi, bir ODBC veya OLE DB bağlantısı aracılığıyla rastgele bir SQL komutu göndermenize olanak tanır.

Söz Dizimi:

```
SQL sql_command
```

Veritabanını güncelleyen SQL deyimleri gönderildiğinde, Qlik Sense uygulaması ODBC bağlantısını salt okunur modda açarsa bir hata döndürülür.

Söz dizimi:

```
SQL SELECT * from tab1;
```

Bu söz dizimine izin verilir ve tutarlılık sağlamak amacıyla **SELECT** için tercih edilen söz dizimi budur. Ancak SQL öneki **SELECT** deyimleri için isteğe bağlı nitelikte kalır.

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
sql_command	Geçerli bir SQL komutu.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

sqlcolumns deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının sütunlarını açıklayan bir alan setini döndürür.

Söz Dizimi:

```
SQLcolumns
```

Bu alanlar, belirli bir veritabanına yönelik iyi bir genel bakış sağlamak için **sqltables** ve **sqltypes** komutlarının oluşturduğu alanlarla birleştirilebilir. On iki standart alan şunlardır:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Bu alanların ayrıntılı açıklaması için ODBC referans el kitabına bakın.

Örnek:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Bazı ODBC sürücüleri bu komut desteklemeyebilir. Bazı ODBC sürücüleri ek alanlar üretebilir.

SQLTables

sqltables deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının tablolarını açıklayan bir alan setini döndürür.

Söz Dizimi:

```
SQLTables
```

Bu alanlar, belirli bir veritabanına yönelik iyi bir genel bakış sağlamak için **sqlcolumns** ve **sqltypes** komutlarının oluşturduğu alanlarla birleştirilebilir. Beş standart alan şunlardır:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Bu alanların ayrıntılı açıklaması için ODBC referans el kitabına bakın.

Örnek:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Bazı ODBC sürücüleri bu komut desteklemeyebilir. Bazı ODBC sürücüleri ek alanlar üretebilir.

SQLTypes

sqltypes deyimi, **connect** yapılmış bir ODBC veya OLE DB veri kaynağının türlerini açıklayan bir alan setini döndürür.

Söz Dizimi:

```
SQLTypes
```

Bu alanlar, belirli bir veritabanına yönelik iyi bir genel bakış sağlamak için **sqlcolumns** ve **sqltables** komutlarının oluşturduğu alanlarla birleştirilebilir. On beş standart alan şunlardır:

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

Bu alanların ayrıntılı açıklaması için ODBC referans el kitabına bakın.

Örnek:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Bazı ODBC sürücüleri bu komut desteklemeyebilir. Bazı ODBC sürücüleri ek alanlar üretebilir.

Star

Veritabanındaki bir alanın tüm değerler kümesini temsilen kullanılan dize **star** deyimi aracılığıyla ayarlanabilir. Sonrasında gelen **LOAD** ve **SELECT** deyimlerini etkiler.

Söz Dizimi:

```
Star is[ string ]
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	Rastgele seçilen bir metin. Boşluklar içermesi durumunda dizenin tırnak işaretleri içine alınması gerektiğini unutmayın. Hiçbir şey belirtilmezse **star is;** olduğu varsayılır; yani açıkça belirtilmediği takdirde kullanılabilecek bir yıldız sembolü yoktur. Bu tanım yeni bir **star** deyimi belirtilene kadar geçerlidir.

Bölüm erişimi kullanılıyorsa, komut dosyasının veri kısmında (**Bölüm Uygulaması** bölümünde) **Star is** deyiminin kullanılması önerilmez. Ancak komut dosyasının **Bölüm Erişimi** kısmında korumalı alanlar için yıldız karakteri tamamen desteklenir. Bu durumda, her zaman bölüm erişiminde örtük olduğundan belirtik **Star is** deyimini kullanmanız gerekmez.

Sınırlamalar

- Anahtar alanlarla, başka bir deyişle tabloları bağlayan alanlarla yıldız karakterini kullanamazsınız.
- Tabloları bağlayan alanları etkileyebileceğinden, **Unqualify** deyiminden etkilenen alanlarla yıldız karakterini kullanamazsınız.
- Mantıksal olmayan tablolarla (örneğin, bilgi yükü tabloları veya eşleme yükü tabloları ile) yıldız karakterini kullanamazsınız.
- Bölüm erişiminde bir azaltma alanında (verilere bağlanan bir alan) yıldız karakteri kullanıldığında bu, bölüm erişiminde bu alanda listelenen değerleri temsil eder. Verilerde mevcut olabilecek, ancak bölüm erişiminde listelenmeyen diğer değerleri temsil etmez.
- **Bölüm Erişimi** alanının dışında herhangi bir veri azaltma biçiminden etkilenen alanlarla yıldız karakterini kullanamazsınız.

Örnek

Aşağıdaki örnek, bölüm erişimi sunan veri kod dosyasının özetidir.

```
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

];
```

Aşağıdakiler geçerlidir:

- *Star* işareti * olur.
- *ADMIN* kullanıcısı tüm alanları görür. Hiçbir şey çıkarılmaz.
- *USER1* kullanıcısı *SALES* alanını göremez.
- *USER2* kullanıcısı *WAREHOUSE* alanını göremez.
- *USER3* kullanıcısı *EMPLOYEES* alanını göremez.

- *USER4* kullanıcısı, bu kullanıcı için iki alanda (*SALES* ve *WAREHOUSE*) OMIT uygulamak amacıyla iki kez eklenir.
- *USER5* için "*" eklenmiştir; bu, OMIT'te listelenen alanların hiçbirinin kullanılamadığı, yani *USER5* kullanıcısının *SALES*, *WAREHOUSE* ve *EMPLOYEES* alanlarını göremediği, fakat *ORDERS* alanını görebildiği anlamına gelir.

Store

Store deyimi bir QVD, CSV veya text dosyası olşuşturur.

Söz Dizimi:

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

Deyim, açıkça adlandırılmış bir QVD, CSV veya TXT dosyası oluşturur.

Deyim yalnızca bir veri tablosundan alanları dışa aktarabilir. Birkaç tablodan alanlar dışa aktarılacaksa, dışa aktarılması gereken veri tablosunu oluşturmak için kodda önceden açık bir join yapılmalıdır.

Metin değerleri, CSV dosyasına UTF-8 biçiminde dışa aktarılır. Bir sınırlayıcı belirtilebilir, bkz. **LOAD**. Bir CSV dosyasına yönelik **store** deyimi BIFF dışa aktarımı desteklemez.

Bağımsız Değişkenler:

Store komutu bağımsız değişkenleri

Bağımsız Değişken	Açıklama
*fieldlist::= (* | field) { , field })*	Seçilecek alanların listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. *field::= fieldname [***as** *aliasname]* *fieldname*, *table* içindeki bir alan adıyla aynı olan metindir. (Boşluklar veya diğer standart olmayan karakterler içermesi halinde alan adının düz çift tırnak işaretleri veya köşeli ayraçlar içine alınması gerektiğini unutmayın.) *aliasname*, sonuç olarak elde edilen QVD veya CSV dosyasında kullanılacak alan için bir alternatif addır.
table	Veriler için kaynak olarak kullanılacak önceden yüklenmiş bir tabloyu temsil eden kod etiketi.

Bağımsız Değişken	Açıklama
filename	Mevcut bir klasör veri bağlantısının geçerli yolu dahil hedef dosyanın adı. **Örnek:** *'lib://Table Files/target.qvd'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data\sales.qvd* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data\sales.qvd* Yol atlanırsa, Qlik Sense bu dosyayı **Directory** deyiminde belirtilen dizinde saklar. **Directory** deyimi yoksa, Qlik Sense dosyayı *C:\Users\{user}\Documents\Qlik\Sense\Apps* çalışma dizininde depolar.
format-spec ::=((**txt** | **qvd**))	Biçim belirtimi, metin dosyaları için metin **txt** veya **qvd** dosyaları için metin qvd öğesinden oluşur. Biçim belirtimi atlanırsa **qvd** olduğu varsayılır.

Örnekler:

```
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';
```

DataFiles bağlantılarının dosya uzantısı büyük-küçük harfe duyarlıdır. Örneğin: .qvd.

Table/Tables

Table ve **Tables** kod anahtar sözcükleri **Drop**, **Comment** ve **Rename** deyimlerinde ve bunun yanı sıra bir biçim belirticisi olarak **Load** deyimlerinde kullanılır.

Tag

Bu kod deyimi, bir veya daha fazla alana veya tabloya etiket atama yolu sağlar. Uygulamada mevcut olmayan bir alanı veya tabloyu etiketleme girişimi olursa etiketleme yoksayılacaktır. Bir alan veya etiket adının çakışan oluşları varsa, son değer kullanılır.

Söz Dizimi:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fieldlist	Virgülle ayrılmış bir listede etiketlenmesi gereken bir veya birkaç alan.
mapname	Bir **mapping Load** veya **mapping Select** deyiminde daha önce yüklenmiş bir eşleme tablosunun adı.
tablelist	Etiketlenmesi gereken tabloları içeren virgülle ayrılmış liste.
tagname	Alana uygulanması gereken etiketin adı.

Example 1:

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

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

trace deyimi, kullanıldığında, **Kod Yürütme İlerlemesi** penceresine ve kod günlük dosyasına bir dize yazar. Bu deyim, hata ayıklama amaçlı kullanımda çok faydalıdır. **trace** deyimi öncesinde hesaplanan değişkenlerin $ genişletmelerini kullanarak, mesajı özelleştirebilirsiniz.

Söz Dizimi:

```
Trace string
```

Example 1:

Aşağıdaki deyim, "Main" tablosunu yükleyen Load deyiminden hemen sonra kullanılabilir.

```
Trace Main table loaded;
```

Bu, kod yürütme diyalog penceresinde ve günlük dosyasında "Main table loaded" metnini görüntüler.

Example 2:

Aşağıdaki deyimler, "Main" tablosunu yükleyen Load deyiminden hemen sonra kullanılabilir.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Bu, kod yürütme diyalog penceresinde ve günlük dosyasında "265,391 rows in Main table" gibi satır sayısını gösteren bir metin görüntüler.

Unmap

Unmap deyimi, arkasından gelen yüklenmiş alanlar için olan önceki bir **Map … Using** deyimi ile belirlenen alan değeri eşlemesini devre dışı bırakır.

Söz Dizimi:

```
Unmap *fieldlist
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	Kod içinde artık bu noktadan eşlenmemesi gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir.

Örnekler ve sonuçlar:

Örnek	Sonuç
Unmap Country;	Country alanının eşlemesini devre dışı bırakır.
Unmap A, B, C;	A, B ve C alanlarının eşlemesini devre dışı bırakır.
Unmap * ;	Tüm alanların eşlemesini devre dışı bırakır.

Unqualify

Unqualify deyimi, daha önce **Qualify** deyimiyle açılmış olan alan adlarının nitelenmesini kapatmak için kullanılır.

Söz Dizimi:

```
Unqualify *fieldlist
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
*fieldlist	Nitelemenin açılması gereken alanların virgülle ayrılmış listesi. Alan listesi olarak * kullanılması tüm alanlara işaret eder. Alan adlarında * ve ? joker karakterlerine izin verilir. Joker karakterler kullanıldığında alan adlarının tırnak içine alınması gerekebilir. Daha fazla bilgi için **Qualify** deyimi belgelerine başvurun.

Example 1:

Alışık olunmayan bir veritabanında, aşağıdaki örnekte gösterildiği gibi, yalnızca bir veya birkaç alanın ilişkilendirildiğinden emin olarak başlamak çoğunlukla faydalı olur:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

İlk olarak tüm alanlar için niteleme açılır.
Sonra **TransID** için niteleme kapatılır.
tab1, *tab2* ve *tab3* tabloları arasındaki ilişkilendirmeler için yalnızca **TransID** alanı kullanılır. Diğer tüm alanlar, tablo adıyla nitelendirilir.

Untag

Bu kod deyimi, alan veya tablolardan etiket kaldırma yolu sağlar. Uygulamada mevcut olmayan bir alandan veya tablodan etiket kaldırma girişimi olursa etiket kaldırma yoksayılacaktır.

Söz Dizimi:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fieldlist	Virgülle ayrılmış bir listede etiketlerin kaldırılması gereken bir veya birkaç alan.
mapname	Bir eşleme **LOAD** veya eşleme **SELECT** deyiminde daha önce yüklenmiş bir eşleme tablosunun adı.
tablelist	Etiketi kaldırılması gereken tabloları içeren virgülle ayrılmış liste.
tagname	Alandan kaldırılması gereken etiketin adı.

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 Çalışma dizini

Kod deyiminde bir dosyaya referansta bulunuyorsanız ve dosya yolu atlanırsa, Qlik Sense dosyayı şu sıraya göre arar:

1. **Directory** deyimi tarafından belirtilen dizin (yalnızca eski kod oluşturma modunda desteklenir).
2. **Directory** deyimi yoksa, Qlik Sense dosyayı çalışma dizininde arar.

Qlik Sense Desktop çalışma dizini

Qlik Sense Desktop içinde çalışma dizini şudur: *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Qlik Sense çalışma dizini

Qlik Sense sunucu yüklemesinde, çalışma dizini Qlik Sense Repository Serviceiçinde belirtilir, varsayılan olarak *C:\ProgramData\Qlik\Sense\Apps*'tir. Daha fazla bilgi için Qlik Management Console yardımına bakın.

2 Veri yükleme düzenleyicisinde değişkenlerle çalışma

Qlik Sense içindeki bir değişken, sayısal veya alfasayısal bir değer gibi statik bir değeri ya da hesaplamayı depolayan bir konteynerdir. Uygulamadaki değişkeni kullandığınızda, değişkende yaptığınız değişiklikler değişkenin kullanıldığı her yerde uygulanır. Değişkenleri, değişkenlere genel bakışta veya veri yükleme düzenleyicisini kullanarak kodda tanımlayabilirsiniz. Bir değişkenin değerini, veri yükleme komut dosyasındaki **Let** ve **Set** deyimlerini kullanarak ayarlarsınız.

Bir sayfayı düzenlerken değişkenlere genel bakıştan Qlik Sense değişkenleriyle de çalışabilirsiniz.

2.7 Genel Bakış

Bir değişken değerinin ilk karakteri '=' eşittir işaretiyse, Qlik Sense, değerleri formül (Qlik Sense ifadesi) olarak değerlendirmeye ve ardından asıl formül metni yerine sonucu görüntülemeye veya döndürmeye çalışır.

Kullanıldığı zaman, değişkenin yerini değişkenin değeri alır. Değişkenler, dolar işareti genişletmesi için kod içinde ve çeşitli kontrol ifadelerinde kullanılabilir. Bu, aynı dizenin kod içinde birçok kez tekrarlanması durumda (örneğin bir yol için) çok kullanışlı olur.

Bazı özel sistem değişkenleri, önceki değerlerine bakılmaksızın kod yürütmesinin başlangıcında Qlik Sense tarafından ayarlanır.

2.8 Bir değişkeni tanımlama

Değişkenler, statik değerleri veya bir hesaplamanın sonucunu saklama olanağı sağlar. Bir değişken tanımlarken aşağıdaki söz dizimini kullanın:

```
set variablename = string
```

veya

```
let variable = expression
```

Set deyimi dize ataması için kullanılır. Eşittir işaretinin sağındaki metni değişkene atar. **Let** deyimi, kod çalıştırma zamanında eşittir işaretinin sağındaki bir ifadeyi değerlendirir ve ifadenin sonucunu değişkene atar.

Değişkenler büyük/küçük harf duyarlıdır.

Qlik Sense içindeki bir alanda veya fonksiyonda bir değişkene aynı adı vermek önerilmez.

Örnekler:

`set x = 3 + 4; //` değişkeni değer olarak '3 + 4' dizesini alır.

`let x = 3 + 4; //`, değer olarak 7'yi döndürür.

`set x = Today(); //`, değer olarak 'Today()' ifdesini döndürür.

`let x = Today(); //`, değer olarak bugünün tarihini döndürür, örneğin, '9/27/2021'.

2.9 Bir değişkeni silme

Koddan bir değişkeni kaldırıp verileri yeniden yüklerseniz değişken uygulamada kalır. Değişkeni uygulamadan tamamen kaldırmak istiyorsanız değişkeni değişkenler diyalog penceresinden de silmelisiniz.

2.10 Değişken değerini alan değeri olarak yükleme

LOAD deyiminde alan değeri olarak bir değişken değerini yüklemek isterseniz ve dolar genişletmesinin sonucu sayı veya ifade yerine metin olursa, genişletilmiş değişkeni tek tırnak içine almanız gerekir.

Örnek:

Bu örnek, kod hatalarının listesini içeren sistem değişkenini bir tabloya yükler. **If** cümlesindeki ScriptErrorCount genişletmesinin tırnak işareti gerektirmediğini, ScriptErrorList genişletmesinin ise gerektirdiğini görebilirsiniz.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1; END IF
```

2.11 Değişken hesaplaması

Qlik Sense uygulamasında hesaplanan değerler ile değişkenleri kullanmanın çeşitli yolları vardır ve bunu nasıl tanımladığınıza ve ifade içinde nasıl çağırdığınıza göre sonuç değişir.

Bu örnekte, bazı satır içi veriler yüklüyoruz:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

İki değişken tanımlayalım:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

İkinci değişkende ifadenin önüne bir eşittir işareti ekliyoruz. Böylece değişkenin, genişletme yapılmadan ve ifade değerlendirilmeden önce hesaplanması sağlanır.

vSales değişkenini olduğu gibi kullanırsanız (örneğin, bir hesaplama içinde) sonuç Sum(Sales) dizesi olur; yani hiçbir hesaplama yapılmaz.

Dolar işareti genişletmesi ekler ve $(vSales) öğesini ifade içinde çağırırsanız, değişken genişletilir ve Sales toplamı görüntülenir.

Son olarak, $(vSales2) öğesini çağırırsanız değişken genişletilmeden önce hesaplanır. Bu da görüntülenen sonucun Sales toplamı olması anlamına gelir. Hesaplama ifadesi olarak =$(vSales) ile =$(vSales2) kullanma arasındaki fark, sonuçların gösterildiği bu grafikte görülmektedir:

Sonuçlar

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Göreceğiniz üzere $(vSales) sonuç olarak bir boyut değerinin kısmi toplamını verirken, $(vSales2) sonuç olarak genel toplamı verir.

Aşağıdaki kod değişkenleri kullanılabilir durumdadır:

- *Hata değişkenleri (page 163)*
- *Sayı yorumlama değişkenleri (page 149)*
- *Sistem değişkenleri (page 141)*
- *Değer işleme değişkenleri (page 147)*

2.12 Sistem değişkenleri

Bazıları sistem tanımlı olan sistem değişkenleri sistem ve Qlik Sense uygulaması hakkında bilgiler sağlar.

Sistem değişkenlerine genel bakış

Genel bakıştan sonra bazı fonksiyonlar daha ayrıntılı olarak açıklanmaktadır. Bu fonksiyonlar için, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Floppy

Bulunan ilk disket sürücüsünün sürücü harfini döndürür; bu normalde *a:* şeklindedir. Bu, sistem tanımlı bir değişkendir.

```
Floppy
```

Bu değişken, standart modda desteklenmez.

CD

Bulunan ilk CD-ROM sürücüsünün sürücü harfini döndürür. CD-ROM bulunmazsa *c:* döndürülür. Bu, sistem tanımlı bir değişkendir.

```
CD
```

Bu değişken, standart modda desteklenmez.

Include

Include/Must_Include değişkeni, koda eklenmesi ve kod olarak değerlendirilmesi gereken metni içeren bir dosyayı belirtir. Veri eklemek için kullanılmaz. Kodunuzun bölümlerinizi ayrı bir metin dosyasında depolayabilir ve birkaç uygulamada yeniden kullanabilirsiniz. Bu, kullanıcı tanımlı bir değişkendir.

```
$(Include=filename)
$(Must_Include=filename)
```

HidePrefix

Bu metin dizesiyle başlayan tüm alan adları, sistem alanlarının gizlendiği şekilde gizlenir. Bu, kullanıcı tanımlı bir değişkendir.

```
HidePrefix
```

HideSuffix

Bu metin dizesiyle biten tüm alan adları, sistem alanlarının gizlendiği şekilde gizlenir. Bu, kullanıcı tanımlı bir değişkendir.

```
HideSuffix
```

QvPath

Qlik Sense yürütülebilir dosyasına yönelik gözatma dizesini döndürür. Bu, sistem tanımlı bir değişkendir.

```
QvPath
```

Bu değişken, standart modda desteklenmez.

QvRoot

Qlik Sense yürütülebilir dosyasının kök dizinini döndürür. Bu, sistem tanımlı bir değişkendir.

```
QvRoot
```

Bu değişken, standart modda desteklenmez.

QvWorkPath

Geçerli Qlik Sense uygulamasına yönelik gözatma dizesini döndürür. Bu, sistem tanımlı bir değişkendir.

```
QvWorkPath
```

Bu değişken, standart modda desteklenmez.

QvWorkRoot

Geçerli Qlik Sense uygulamasının kök dizinini döndürür. Bu, sistem tanımlı bir değişkendir.

```
QvWorkRoot
```

Bu değişken, standart modda desteklenmez.

StripComments

Bu değişken 0 olarak ayarlanırsa, kodda /*..*/ ve // yorumlarına yönelik arındırma işlemi yasaklanır. Bu değişken tanımlanmazsa, yorumların arındırılması her zaman gerçekleştirilir.

```
StripComments
```

Verbatim

Normalde tüm alan değerleri, Qlik Sense veritabanına yüklenmeden önce öndeki ve sondaki boşluklardan (ASCII 32) otomatik olarak arındırılır. Bu değişkenin 1 olarak ayarlandığında, boşluklara yönelik arındırma işlemi askıya alınır. Sekme (ASCII 9) ve bölünemez boşluk (ANSI 160) karakterleri asla arındırılmaz.

```
Verbatim
```

OpenUrlTimeout

Bu değişken, Qlik Sense uygulamasının URL kaynaklarından (örn. sayfalardan) veri alırken uyması gereken zaman aşımını saniye cinsinden HTML tanımlar. Atlandığı takdirde zaman aşımı yaklaşık 20 dakika olur.

```
OpenUrlTimeout
```

WinPath

Windows'a yönelik gözatma dizesini döndürür. Bu, sistem tanımlı bir değişkendir.

```
WinPath
```

Bu değişken, standart modda desteklenmez.

WinRoot

Windows'un kök dizinini döndürür. Bu, sistem tanımlı bir değişkendir.

```
WinRoot
```

Bu değişken, standart modda desteklenmez.

CollationLocale
Sıralama düzeni ve arama eşleşmesi için hangi yerel ayarın kullanılacağını belirtir. Değer, bir yerel ayarın kültür adıdır (örneğin, 'en-US').Bu, sistem tanımlı bir değişkendir.

```
CollationLocale
```

CreateSearchIndexOnReload
Bu değişken, arama indeksi dosyalarının veriler yeniden yüklendiği sırada oluşturulması gerekip gerekmediğini tanımlar.

```
CreateSearchIndexOnReload
```

CreateSearchIndexOnReload

Bu değişken, arama indeksi dosyalarının veriler yeniden yüklendiği sırada oluşturulması gerekip gerekmediğini tanımlar.

Söz Dizimi:

```
CreateSearchIndexOnReload
```

Arama dizin dosyalarının veriler yeniden yüklendiği sırada mı yoksa kullanıcının ilk arama isteğinden sonra mı oluşturulacağını tanımlayabilirsiniz. Veriler yeniden yüklendiği sırada arama dizini oluşturmanın avantajı, ilk kullanıcı bir arama yaptığında yaşanan gecikme süresinden kaçınmaktır. Bunun arama dizini oluşturma için gereken verileri yeniden yükleme süresiyle karşılaştırılarak değerlendirilmesi gerekir.

Bu değişken atlanırsa arama dizini dosyaları veriler yeniden yüklendiği sırada oluşturulmaz.

Oturum uygulamaları için bu değişkenin ayarından bağımsız olarak arama dizini dosyaları veriler yeniden yüklendiği sırada oluşturulmaz.

Example 1: Arama dizini dosyalarını veriler yeniden yüklendiği sırada oluştur

```
set CreateSearchIndexOnReload=1;
```

Example 2: Arama dizini dosyalarını ilk arama isteğinden sonra oluştur

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Bu metin dizesiyle başlayan tüm alan adları, sistem alanlarının gizlendiği şekilde gizlenir. Bu, kullanıcı tanımlı bir değişkendir.

Söz Dizimi:

```
HidePrefix
```

Örnek:

```
set HidePrefix='_' ;
```

Bu deyim kullanılırsa, sistem alanları gizlendiğinde alt çizgiyle başlayan alan adları, alan adları listesinde gösterilmez.

HideSuffix

Bu metin dizesiyle biten tüm alan adları, sistem alanlarının gizlendiği şekilde gizlenir. Bu, kullanıcı tanımlı bir değişkendir.

Söz Dizimi:

```
HideSuffix
```

Örnek:

```
set HideSuffix='%';
```

Bu deyim kullanılırsa, sistem alanları gizlendiğinde yüzde işaretiyle biten alan adları, alan adları listesinde gösterilmez.

Include

Include/Must_Include değişkeni, koda eklenmesi ve kod olarak değerlendirilmesi gereken metni içeren bir dosyayı belirtir. Veri eklemek için kullanılmaz. Kodunuzun bölümlerinizi ayrı bir metin dosyasında depolayabilir ve birkaç uygulamada yeniden kullanabilirsiniz. Bu, kullanıcı tanımlı bir değişkendir.

Bu değişken, yalnızca standart modda klasör veri bağlantılarını destekler.

Söz Dizimi:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Değişkenin iki sürümü vardır:

- **Include**, dosya bulunamadığı takdirde bir hata üretmez ve sessizce başarısız olur.
- **Must_Include** ise dosya bulunamadığı takdirde hata üretir.

Bir yol belirtmezseniz, dosya adı Qlik Sense uygulaması çalışma dizinine göreceli olur. Mutlak dosya yolu veya lib:// klasör bağlantısının yolunu da belirtebilirsiniz. Eşittir işaretinden önce ve sonra boşluk karakteri koymayın.

set Include =*filename yapısı uygulanamaz.*

Örnekler:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Sınırlamalar

Windows ve Linux altında UTF-8 kodlu dosyalar arasında sınırlı çapraz uyumluluk.

UTF-8'i BOM (Bayt Sırası İşareti) ile kullanmak isteğe bağlıdır. BOM, bir dosyanın başlangıcında ASCII olmayan baytlar beklemeyen, ancak metin akışını işleyebilecek yazılımlarda UTF-8 kullanımına müdahale edebilir.

- Windows sistemleri, bayt depolamasında belirsizlik olmamasına rağmen, bir dosyanın UTF-8 kodlu olduğunu belirlemek için UTF-8'de BOM kullanır.
- Unix / Linux, Unicode için UTF-8 kullanır, ancak BOM'u kullanmaz çünkü bu, komut dosyalarının sözdizimini bozar.

Bunu Qlik Sense için bazı etkileri vardır.

- Windows'ta UTF-8 BOM ile başlayan herhangi bir dosya UTF-8 komut dosyası olarak kabul edilir. Aksi takdirde ANSI kodlaması varsayılır.
- Linux'ta, sistem varsayılan 8 bit kod sayfası UTF-8'dir. Bu nedenle UTF-8 bir BOM içermemesine rağmen çalışır.

Sonuç olarak taşınabilirlik garanti edilemez. Windows'ta Linux tarafından yorumlanabilen (veya tersi olan) bir dosya oluşturmak her zaman mümkün değildir. BOM'un farklı işlenmesi nedeniyle, UTF-8 kodlu dosyalara ilişkin iki sistem arasında çapraz uyumluluk yoktur.

OpenUrlTimeout

Bu değişken, Qlik Sense uygulamasının URL kaynaklarından (örn. sayfalardan) veri alırken uyması gereken zaman aşımını saniye cinsinden HTML tanımlar. Atlandığı takdirde zaman aşımı yaklaşık 20 dakika olur.

Söz Dizimi:

```
OpenUrlTimeout
```

Örnek:

```
set OpenUrlTimeout=10;
```

StripComments

Bu değişken 0 olarak ayarlanırsa, kodda /*..*/ ve // yorumlarına yönelik arındırma işlemi yasaklanır. Bu değişken tanımlanmazsa, yorumların arındırılması her zaman gerçekleştirilir.

Söz Dizimi:

```
StripComments
```

Belirli veritabanı sürücüleri, **SELECT** deyimlerinde optimizasyon ipuçları olarak /*..*/ kullanır. Böyle bir durum söz konusu ise, **SELECT** deyimi veritabanı sürücüsüne gönderilmeden önce yorumlar arındırılmamalıdır.

Gerektiğinde, bu değişkenin deyimlerden hemen sonra 1'e sıfırlanması önerilir.

Örnek:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Normalde tüm alan değerleri, Qlik Sense veritabanına yüklenmeden önce öndeki ve sondaki boşluklardan (ASCII 32) otomatik olarak arındırılır. Bu değişkenin 1 olarak ayarlandığında, boşluklara yönelik arındırma işlemi askıya alınır. Sekme (ASCII 9) ve bölünemez boşluk (ANSI 160) karakterleri asla arındırılmaz.

Söz Dizimi:

```
Verbatim
```

Örnek:

```
set Verbatim = 1;
```

2.13 Değer işleme değişkenleri

Bu bölümde, NULL ve diğer değerleri işlemek için kullanılan değişkenler açıklanmaktadır.

Değer işleme değişkenlerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

NullDisplay
Tanımlanmış sembol, verilerin en düşük değerinde ODBC'den gelen tüm NULL değerlerini ve bağlayıcıları ikame eder. Bu, kullanıcı tanımlı bir değişkendir.

```
NullDisplay
```

NullInterpret
Bu tanımlanmış sembol bir metin dosyası, Excel dosyası veya satır içi deyimi içinde geçtiğinde NULL olarak yorumlanacaktır. Bu, kullanıcı tanımlı bir değişkendir.

```
NullInterpret
```

NullValue
NullAsValue deyimi kullanılırsa, tanımlanan sembol, **NullAsValue** belirtilen alanlarındaki tüm NULL değerleri belirtilen dize ile değiştirir.

```
NullValue
```

OtherSymbol
Bir **LOAD/SELECT** deyimi öncesinde 'tüm diğer değerler' olarak işlenecek bir sembolü tanımlar. Bu, kullanıcı tanımlı bir değişkendir.

```
OtherSymbol
```

NullDisplay

Tanımlanmış sembol, verilerin en düşük değerinde ODBC'den gelen tüm NULL değerlerini ve bağlayıcıları ikame eder. Bu, kullanıcı tanımlı bir değişkendir.

Söz Dizimi:

```
NullDisplay
```

Örnek:

```
set NullDisplay='<NULL>';
```

NullInterpret

Bu tanımlanmış sembol bir metin dosyası, Excel dosyası veya satır içi deyimi içinde geçtiğinde NULL olarak yorumlanacaktır. Bu, kullanıcı tanımlı bir değişkendir.

Söz Dizimi:

```
NullInterpret
```

Örnekler:

```
set NullInterpret=' ';
set NullInterpret =;
```

Excel'de boş değerler için NULL değerler döndürmez, ancak CSV metin dosyasında döndürür.

```
set NullInterpret ='';
```

Excel'de boş değerler için NULL değerler döndürür.

NullValue

NullAsValue deyimi kullanılırsa, tanımlanan sembol, **NullAsValue** belirtilen alanlarındaki tüm NULL değerleri belirtilen dize ile değiştirir.

Söz Dizimi:

```
NullValue
```

Örnek:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Bir **LOAD/SELECT** deyimi öncesinde 'tüm diğer değerler' olarak işlenecek bir sembolü tanımlar. Bu, kullanıcı tanımlı bir değişkendir.

Söz Dizimi:

```
OtherSymbol
```

Örnek:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

Y='b' alan değeri artık, diğer sembol üzerinden Z='c' öğesine bağlanır.

2.14 Sayı yorumlama değişkenleri

Sayı yorumlama değişkenleri sistem tanımlıdır; yani, yeni bir uygulama oluşturulduğunda işletim sisteminin geçerli bölgesel ayarlarına göre otomatik olarak oluşturulurlar. Qlik Sense Desktop ortamında bu, bilgisayar işletim sisteminin ayarlarına göre yapılır ve Qlik Sense ortamında ise Qlik Sense uygulamasının yüklü olduğu sunucunun işletim sistemine göre yapılır.

Değişkenler, yeni Qlik Sense uygulaması kodunun üst kısmına eklenir ve kod yürütme sırasında belirli sayı biçimlendirme ayarları için işletim sistemi varsayılanlarının yerini alır. Bunlar rahatlıkla silinebilir, düzenlenebilir veya çoğaltılabilir.

Bir uygulamayı belirli bir yer için oluşturmak isterseniz bunu yapmanın en kolay you muhtemelen uygulamayı oluşturmak için Qlik Sense Desktop uygulamasını, işletim sisteminde istenilen yer ayarına sahip bir bilgisayarda kullanmaktır. Uygulama o yere yönelik uygun bölgesel ayarları içerir ve daha fazla geliştirmek için uygulamayı tercih ettiğiniz bir Qlik Sense sunucusuna taşıyabilirsiniz.

Sayı yorumlama değişkenlerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Para birimi biçimlendirmesi

MoneyDecimalSep

Tanımlanmış ondalık ayırıcı işletim sisteminin para birimi ondalık sembolünün yerini alınır.

```
MoneyDecimalSep
```

MoneyFormat

Tanımlanmış sembol işletim sisteminin para birimi sembolünün yerini alır.

```
MoneyFormat
```

MoneyThousandSep

Tanımlanmış binlik ayracı işletim sisteminin para birimi basamak gruplandırma sembolünün yerini alır.

```
MoneyThousandSep
```

Sayı biçimlendirme

DecimalSep

Tanımlanmış ondalık ayırıcı işletim sisteminin (bölgesel ayarlar) ondalık sembolünün yerini alır.

```
DecimalSep
```

ThousandSep

Tanımlanmış binlik ayracı işletim sisteminin basamak gruplandırma sembolünün yerini alır.

```
ThousandSep
```

NumericalAbbreviation

Sayısal kısaltmalar, sayıların ölçek önekleri için hangi kısaltmanın kullanılacağını ayarlar, örneğin mega veya milyon için M (10^6), ve mikro için µ (10^{-6}).

```
NumericalAbbreviation
```

Zaman biçimlendirmesi

DateFormat

Bu ortam değişkeni, uygulamada varsayılan olarak kullanılan tarih biçimini tanımlar. Biçim, tarihleri hem yorumlamak hem biçimlendirmek için kullanılır. Değişken tanımlı değilse, kod çalıştırıldığında işletim sisteminin bölgesel ayarlarının tarih biçimi getirilir.

```
DateFormat
```

TimeFormat

Tanımlanmış biçim işletim sisteminin zaman biçiminin yerini alınır.

```
TimeFormat
```

TimestampFormat

Tanımlanmış biçim işletim sisteminin tarih ve zaman biçiminin yerini alır.

```
TimestampFormat
```

MonthNames

Tanımlanmış biçim işletim sisteminin ay adları kurallarının yerini alır.

```
MonthNames
```

LongMonthNames

Tanımlanmış biçim işletim sisteminin ay uzun adları kurallarının yerini alır.

```
LongMonthNames
```

DayNames

Tanımlanmış biçim işletim sisteminin (bölgesel ayarlar) haftanın günleri kurallarının yerini alınır.

```
DayNames
```

LongDayNames

Tanımlanmış biçim işletim sisteminin haftanın günlerinin uzun adları kurallarının yerini alınır.

```
LongDayNames
```

FirstWeekDay

Haftanın ilk günü olarak hangi günün kullanılacağını tanımlayan tamsayı.

```
FirstWeekDay
```

BrokenWeeks

Bu ayar, haftaların bölünüp bölünmeyeceğini tanımlar.

```
BrokenWeeks
```

ReferenceDay

Ayar, Ocak ayında hangi günün 1. haftayı tanımlamak için referans gün olarak ayarlanacağını tanımlar.

ReferenceDay

FirstMonthOfYear
Ayar, yılın ilk ayı olarak hangi ayın kullanılacağını tanımlar. Bu da aylık kaydırma kullanılan mali yılları (örneğin, 1 Nisan ile başlayan) tanımlamak için kullanılabilir.

Bu ayar şu anda kullanılmamaktadır, ancak gelecekte kullanılması planlanmaktadır.

Geçerli ayarlar 1 (Ocak) ila 12 (Aralık) şeklindedir. Varsayılan ayar 1'dir.

Söz Dizimi:
FirstMonthOfYear

Örnek:

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

BrokenWeeks

Bu ayar, haftaların bölünüp bölünmeyeceğini tanımlar.

Söz Dizimi:
BrokenWeeks
Varsayılan olarak, Qlik Sense fonksiyonları bölünmemiş haftaları kullanır. Bunun anlamı şudur:

- Bazı yıllarda 1. hafta Aralık ayı içinde başlar ve bazı yıllarda 52. veya 53. hafta Ocak ayına devam eder.
- 1. haftanın Ocak ayı içinde her zaman en az 4 günü vardır.

Bunun alternatifi bölünmüş haftaları kullanmaktır:

- 52. veya 53. hafta Ocak ayına devam etmez.
- 1. hafta 1 Ocak'tan itibaren başlar ve çoğu durumda tam bir hafta değildir.

Aşağıdaki değerler kullanılabilir:

- 0 (=bölünmemiş haftaları kullan)
- 1 (= bölünmüş haftaları kullan)

Örnekler:

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

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

DateFormat

Bu ortam değişkeni, uygulamada varsayılan olarak kullanılan tarih biçimini tanımlar. Biçim, tarihleri hem yorumlamak hem biçimlendirmek için kullanılır. Değişken tanımlı değilse, kod çalıştırıldığında işletim sisteminin bölgesel ayarlarının tarih biçimi getirilir.

Söz Dizimi:

```
DateFormat
```

Örnekler:

```
Set DateFormat='M/D/YY'; //(US format)

Set DateFormat='DD/MM/YY'; //(UK date format)

Set DateFormat='YYYY-MM-DD'; //(ISO date format)
```

DayNames

Tanımlanmış biçim işletim sisteminin (bölgesel ayarlar) haftanın günleri kurallarının yerini alınır.

Söz Dizimi:

```
DayNames
```

Örnek:

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

DecimalSep

Tanımlanmış ondalık ayırıcı işletim sisteminin (bölgesel ayarlar) ondalık sembolünün yerini alır.

Söz Dizimi:

```
DecimalSep
```

Örnekler:

```
Set DecimalSep='.';

Set DecimalSep=',';
```

FirstWeekDay

Haftanın ilk günü olarak hangi günün kullanılacağını tanımlayan tamsayı.

Söz Dizimi:

```
FirstWeekDay
```

Varsayılan olarak Qlik Sense sistem değişkenleri `FirstWeekDay=6` değerini tanımlar. Bu, Pazar'ın haftanın ilk günü olduğu anlamına gelir.

FirstWeekDay için ayarlanabilecek değerler

Değer	Gün
0	Pazartesi

Değer	Gün
1	Salı
2	Çarşamba
3	Perşembe
4	Cuma
5	Cumartesi
6	Pazar

LongDayNames

Tanımlanmış biçim işletim sisteminin haftanın günlerinin uzun adları kurallarının yerini alınır.

Söz Dizimi:

```
LongDayNames
```

Örnek:

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

LongMonthNames

Tanımlanmış biçim işletim sisteminin ay uzun adları kurallarının yerini alır.

Söz Dizimi:

```
LongMonthNames
```

Örnek:

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

MoneyDecimalSep

Tanımlanmış ondalık ayırıcı işletim sisteminin para birimi ondalık sembolünün yerini alınır.

Söz Dizimi:

```
MoneyDecimalSep
```

Örnek:

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

MoneyFormat

Tanımlanmış sembol işletim sisteminin para birimi sembolünün yerini alır.

Söz Dizimi:

```
MoneyFormat
```

Örnek:

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

MoneyThousandSep

Tanımlanmış binlik ayracı işletim sisteminin para birimi basamak gruplandırma sembolünün yerini alır.

Söz Dizimi:

```
MoneyThousandSep
```

Örnek:

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

MonthNames

Tanımlanmış biçim işletim sisteminin ay adları kurallarının yerini alır.

Söz Dizimi:

```
MonthNames
```

Örnek:

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

NumericalAbbreviation

Sayısal kısaltmalar, sayıların ölçek önekleri için hangi kısaltmanın kullanılacağını ayarlar, örneğin mega veya milyon için M (10^6), ve mikro için µ (10^{-6}).

Söz Dizimi:

```
NumericalAbbreviation
```

`NumericalAbbreviation` değişkenini, noktalı virgülle ayrılmış kısaltma tanımı çiftleri listesini içeren bir dizeye ayarlarsınız. Her bir kısaltma tanımı çifti ölçeği (ondalık tabandaki üs) ve iki nokta üst üste işareti ile ayrılan kısaltmayı içermelidir. Örneğin milyon için `6:M`.

Varsayılan ayar şöyledir: `'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'`.

Örnekler:

Bu ayar bine yönelik öneki t ile ve milyara yönelik öneki B ile değiştirir. t$, M$ ve B$ gibi kısaltmaların olduğu finansal uygulamalar için faydalıdır.

```
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

Ayar, Ocak ayında hangi günün 1. haftayı tanımlamak için referans gün olarak ayarlanacağını tanımlar.

Söz Dizimi:

```
ReferenceDay
```

Varsayılan olarak, Qlik Sense fonksiyonları referans gün olarak 4 kullanır. Bu da 1. haftanın 4 Ocak gününü içermesi gerektiği veya başka bir deyişle 1. haftanın Ocak ayında her zaman en az 4 günü olması gerektiği anlamına gelir.

Farklı bir referans gün ayarlamak için aşağıdaki değerler kullanılabilir:

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

Örnekler:

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

ThousandSep

Tanımlanmış binlik ayracı işletim sisteminin basamak gruplandırma sembolünün yerini alır.

Söz Dizimi:

```
ThousandSep
```

Örnekler:

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

Set ThousandSep=' ';
```

TimeFormat

Tanımlanmış biçim işletim sisteminin zaman biçiminin yerini alınır.

Söz Dizimi:

```
TimeFormat
```

Örnek:

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

TimestampFormat

Tanımlanmış biçim işletim sisteminin tarih ve zaman biçiminin yerini alır.

Söz Dizimi:

```
TimestampFormat
```

Örnek:

Şu örnekler, farklı **SET TimestampFormat** deyimlerinin sonuçlarını göstermek için zaman damgası verileri olarak *1983-12-14T13:15:30Z* kullanır. Kullanılan tarih biçimi **YYYYMMDD**, saat biçimi **h:mm:ss TT** şeklindedir. Tarih biçimi, **SET DateFormat** deyiminde ve saat biçimi ise veri yükleme kodunun en üstünde **SET TimeFormat** deyiminde belirtilir.

Sonuçlar

Örnek	Sonuç
`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

Örnekler: Komut dosyası

Örnek: Yükleme kodu
İlk yükleme kodunda *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'* kullanılır. İkinci yükleme kodunda zaman damgası biçimi *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* olarak değiştirilir. Farklı sonuçlar, **SET TimeFormat** deyiminin farklı saat veri biçimleriyle nasıl çalıştığını gösterir.

Aşağıdaki tabloda, izleyen yükleme kodlarında kullanılan veri kümesi gösterilir. Tablonun ikinci sütununda, veri kümesindeki her bir zaman damgasının biçimi gösterilir. İlk beş zaman damgası, ISO 8601 kurallarını izler, ancak altıncı zaman damgası bu kuralları izlemez.

Veri kümesi

Kullanılan saat verilerini ve veri kümesindeki her bir zaman damgası için biçimi gösteren tablo.

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

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

Yükleme kodu

```
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 ];
```

Sonuçlar

Yükleme kodunda kullanılmakta olan TimestampFormat yorumlama değişkeninin sonuçlarını gösteren Qlik Sense tablosu. Veri kümesindeki son zaman damgası, doğru bir tarih döndürmez.

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	-

Sonraki yükleme kodu aynı veri kümesini kullanır. Ancak altıncı zaman damgasının, ISO 8601 dışındaki biçimiyle eşleşmesi için *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* biçimini kullanır.

Veri yükleme düzenleyicisi'nde önceki örnek kodu aşağıdakiyle değiştirin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

Yükleme kodu

```
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 ];
```

Sonuçlar

Yükleme kodunda kullanılmakta olan TimestampFormat yorumlama değişkeninin sonuçlarını gösteren Qlik Sense tablosu.

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 Direct Discovery değişkenleri

Direct Discovery sistem değişkenleri

DirectCacheSeconds

Önbelleğe alma sınırını görselleştirmeler için Direct Discovery sorgu sonuçlarına göre ayarlayabilirsiniz. Bu süre sınırına erişildikten sonra, Qlik Sense yeni Direct Discovery sorguları yapıldığında önbelleği temizler. Qlik Sense, seçimler için veri kaynağını sorgular ve atanmış süre sınırı için önbelleği yeniden oluşturur. Seçimlerin her bir kombinasyonu için sonuç bağımsız bir şekilde önbelleğe alınır. Yani, önbellek her bir seçim için bağımsız bir şekilde yenilenir; bir seçim yalnızca seçilen alanlar için önbelleği yeniler ve ikinci bir seçim kendi ilgili alanları için önbelleği yeniler. İkinci seçim ilk seçimde yenilenen alanları içermesi halinde, önbellek limitine erişilmemişse bu alanlar önbellekte yeniden güncelleştirilmez.

Direct Discovery önbelleği, **Tablo** görselleştirmelerine uygulanmaz. Tablo seçimleri veri kaynağını her seferinde sorgular.

Sınır değeri saniye olarak ayarlanmalıdır. Varsayılan önbellek sınırı, 1800 saniye (30 dakika) şeklindedir.

DirectCacheSeconds için kullanılan değer, **DIRECT QUERY** deyimi yürütüldüğü anda ayarlanan değerdir. Değer çalışma süresinde değiştirilemez.

Örnek:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Bağlantı havuzu kapasitesini kullanarak veritabanına yönelik olarak asenkron ve paralel çağrılar yapabilirsiniz. Havuz kapasitesini kurmaya yönelik kod dosyası söz dizimi aşağıdaki gibidir:

```
SET DirectConnectionMax=10;
```

Sayısal ayar, Direct Discovery kodunun bir sayfayı güncellerken kullanması gereken veritabanı bağlantılarının maksimum sayısını belirtir. Varsayılan ayar 1 şeklindedir.

Bu değişken dikkatli bir şekilde kullanılmalıdır. 1'den yüksek olarak ayarlandığında Microsoft SQL Server ile bağlantı kurulurken sorunlara yol açtığı bilinmektedir.

DirectUnicodeStrings
Direct Discovery, bazı veritabanlarının (özellikle SQL Server'ın) gerektirdiği şekilde, genişletilmiş karakter düz dizeleri (N'<genişletilmiş dize>') için SQL standart biçimini kullanarak genişletilmiş Unicode verilerin seçimini destekleyebilir. Bu söz diziminin kullanımı, **DirectUnicodeStrings** kod değişkeni ile Direct Discovery için etkinleştirilebilir.

Bu değerin 'true' olarak ayarlanması, düz dizelerin önünde ANSI standart geniş karakter işaretleyicisi “N”in kullanımını mümkün kılar. Tüm veritabanları bu standardı desteklemez. Varsayılan ayar 'false' şeklindedir.

DirectDistinctSupport
DIMENSION alan değeri Qlik Sense nesnesinde seçildiğinde, kaynak veritabanı için sorgu oluşturulur. Sorgu gruplamayı gerektirdiğinde, Direct Discovery yalnızca benzersiz değerleri seçmek için **DISTINCT** anahtar sözcüğünü kullanır. Ancak bazı veritabanları **GROUP BY** anahtar sözcüğünü gerektirir. Benzersiz değerler için sorgularda **DISTINCT** yerine **GROUP BY**'ı oluşturmak için **DirectDistinctSupport**'u 'false' olarak ayarlayın.

```
SET DirectDistinctSupport='false';
```
DirectDistinctSupport true olarak ayarlanırsa, o zaman **DISTINCT** kullanılır. Ayarlanmazsa, varsayılan davranış **DISTINCT** kullanmak olur.

DirectEnableSubquery
Yüksek nicelikte çok tablolu senaryolarda, büyük bir IN cümlesi oluşturmak yerine SQL sorgusunda alt sorgular oluşturulabilir. Bu, **DirectEnableSubquery** 'true' şeklinde ayarlanarak etkinleştirilir. Varsayılan değer 'false' şeklindedir.

***DirectEnableSubquery** etkinleştirildiğinde, Direct Discovery modunda olmayan tabloları yükleyemezsiniz.*

```
SET DirectEnableSubquery='true';
```

Teradata sorgu bantlama değişkenleri

Teradata sorgu bantlama desteği, kurumsal uygulamaların daha iyi muhasebe, önceliklendirme ve iş yükü yönetimi sağlamak amacıyla temel Teradata veritabanıyla işbirliği yapabilmelerini sağlayan bir fonksiyondur. Sorgu bantlamayı kullanarak kullanıcı kimlik bilgileri gibi meta verilerini bir sorgu etrafında kaydırılabilir.

İki değişken mevcut olup, iki dize de değerlendirilir ve veritabanına gönderilir.

SQLSessionPrefix
Bu dize, veritabanıyla bir bağlantı kurulduğunda gönderilir.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

OSuser() örneğin *WA\sbt* döndürürse bu, bağlantı oluşturulduğunda veritabanına gönderilen `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;` olarak değerlendirilecektir.

SQLQueryPrefix
Bu dize her bir ayrı sorgu için gönderilir.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery karakter değişkenleri

DirectFieldColumnDelimiter

Kullanılan karakteri, alan sınırlayıcısı olarak virgül dışında bir karakteri gerektiren veritabanları için **Direct Query** deyimlerinde alan sınırlayıcısı olarak ayarlayabilirsiniz. Belirtilen karakter, **SET** deyiminde tekli tırnak işaretleriyle çevrelenmelidir.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Oluşturulan bir sorguda dizeleri alıntılamak için kullanılacak karakteri belirleyebilirsiniz. Varsayılan, tekli tırnak işaretidir. Belirtilen karakter, **SET** deyiminde tekli tırnak işaretleriyle çevrelenmelidir.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

Oluşturulan sorgularda kullanılacak tanımlayıcıların ANSI olmayan alıntılamaları belirleyebilirsiniz. Şu anda, GoogleBQ'da yalnızca ANSI olmayan alıntılamalar kullanılabilir durumdadır. Varsayılan ANSI'dir. Büyük harf, küçük harf ve büyük-küçük harf karışımı kullanılabilir ((ANSI, ansi, Ansi)).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Örneğin, ANSI alıntılama aşağıdaki **SELECT** deyiminde kullanılır:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteStyle "GoogleBQ" olarak ayarlandığında, **SELECT** deyimi alıntılamayı aşağıdaki gibi kullanır:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Oluşturulan bir sorguda dizeleri tanımlayıcıların alıntılanmasını kontrol edecek karakteri belirleyebilirsiniz. Bu (çift tırnak işareti gibi) tek bir karakter olarak veya (bir çift köşeli ayraç gibi) iki karakter olarak ayarlanabilir. Varsayılan, çift tırnak işaretidir.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold

Direct Discovery alanları **Tablo** görselleştirmesinde kullanıldığında, görüntülenen satır sayısının sınırlandırılması için bir eşik ayarlanır. Varsayılan eşik, 1000 kayıttır. Varsayılan eşik ayarı, kod dosyasında **DirectTableBoxListThreshold** değişkeni ayarlanarak değiştirilebilir. Örneğin:

```
SET DirectTableBoxListThreshold=5000;
```

Eşik ayarı, yalnızca Direct Discovery alanlarını içeren **Tablo** görselleştirmelerine uygulanır. Yalnızca bellek içi alanlar içeren **Tablo** görselleştirmeleri **DirectTableBoxListThreshold** ayarı tarafından sınırlandırılmaz.

Seçim eşik sınırından daha az sayıdaki kayda sahip oluncaya kadar **Tablo** görselleştirmesinde herhangi bir alan görüntülenmez.

Direct Discovery sayı yorumlama değişkenleri

DirectMoneyDecimalSep

Tanımlanmış ondalık ayırıcı, Direct Discovery kullanılarak verileri yüklemek için oluşturulan SQL deyiminde bulunan para biriminin ondalık sembolünün yerini alır. Bu karakter **DirectMoneyFormat** içinde kullanılan karakterle eşleşmelidir.

Varsayılan değer `'.'` şeklindedir.

Örnek:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

Tanımlanmış sembol, Direct Discovery kullanılarak verileri yüklemek için oluşturulan SQL deyiminde bulunan para birimi biçiminin yerini alır. Binlik ayracı için para birimi sembolü dahil edilmemelidir.

Varsayılan değer `'#.0000'` şeklindedir.

Örnek:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

Tanımlanmış zaman biçimi, Direct Discovery kullanılarak verileri yüklemek için oluşturulan SQL deyiminde bulunan zaman biçiminin yerini alır.

Örnek:

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

DirectDateFormat

Tanımlanmış tarih biçimi, Direct Discovery kullanılarak verileri yüklemek için oluşturulan SQL deyiminde bulunan tarih biçiminin yerini alır.

Örnek:

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

DirectTimeStampFormat

Tanımlanmış biçim, Direct Discovery kullanılarak verileri yükleyecek SQL deyiminde oluşturulan SQL deyimindeki tarih ve zaman biçiminin yerini alır.

Örnek:

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

2.16 Hata değişkenleri

Tüm hata değişkenlerinin değerleri kod yürütüldükten sonra mevcut olacaktır. İlk değişken olan ErrorMode kullanıcıdan giriş olarak alınır ve son üç değişken, kod içindeki hatalar hakkında bilgilerle birlikte Qlik Sense uygulamasından çıkış olarak verilir.

Hata değişkenlerine genel bakış

Her değişken genel bakıştan sonra daha ayrıntılı olarak açıklanmaktadır. Söz konusu değişkenin ayrıntılarına anında erişmek için söz dizimindeki değişken adına da tıklayabilirsiniz.

Değişkenler hakkında daha fazla ayrıntı için Qlik Sense çevrimiçi yardımına bakın.

ErrorMode

Bu hata değişkeni, kod yürütmesi sırasında bir hatayla karşılaşıldığında, Qlik Sense tarafından hangi eylemin gerçekleştirileceğini belirler.

```
ErrorMode
```

ScriptError

Bu hata değişkeni, son yürütülen kod deyiminin hata kodunu döndürür.

```
ScriptError
```

ScriptErrorCount

Bu hata değişkeni, geçerli kod yürütmesi sırasında hatalara neden olan deyimlerin toplam sayısını döndürür. Bu değişken kod yürütmesinin başlangıcında her zaman için 0 olarak sıfırlanır.

```
ScriptErrorCount
```

ScriptErrorList

Bu hata değişkeni, son kod yürütmesi sırasında oluşan tüm kod hatalarının birleştirilmiş listesini içerir. Her bir hata, satır beslemesiyle ayrılır.

```
ScriptErrorList
```

ErrorMode

Bu hata değişkeni, kod yürütmesi sırasında bir hatayla karşılaşıldığında, Qlik Sense tarafından hangi eylemin gerçekleştirileceğini belirler.

Söz Dizimi:

```
ErrorMode
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
ErrorMode=1	Varsayılan ayar. Kod yürütmesi durdurulur ve kullanıcıdan eyleme geçmesi istenir (toplu olmayan mod).
ErrorMode =0	Qlik Sense sadece hatayı yok sayar ve bir sonraki kod deyiminde kod yürütmeyi sürdürür.
ErrorMode =2	Qlik Sense, hata olduğunda hemen "Kod yürütme başarısız oldu..." hata mesajını tetikler ve öncesinde kullanıcının eyleme geçmesini istemez.

Örnek:

```
set ErrorMode=0;
```

ScriptError

Bu hata değişkeni, son yürütülen kod deyiminin hata kodunu döndürür.

Söz Dizimi:

```
ScriptError
```

Bu değişken, başarılıyla yürütülen her kod deyiminin ardından 0 olarak sıfırlanır. Hata olursa, dahili bir Qlik Sense hata koduna ayarlanır. Hata kodları, sayı ve metin bileşenlerine sahip ikili değerlerdir. Aşağıdaki hata kodları mevcuttur:

Kod hata kodları

Hata kodu	Açıklama
0	Hata yok. İkili değerli metin boş.
1	Genel hata.
2	Söz dizimi hatası.
3	Genel ODBC hatası.
4	Genel OLE DB hatası.
5	Özel veritabanında genel hata.
6	Genel XML hatası.
7	Genel HTML hatası.

Hata kodu	Açıklama
8	Dosya bulunamadı
9	Veritabanı bulunamadı.
10	Tablo bulunamadı.
11	Alan bulunamadı.
12	Dosyanın biçimi yanlış.
16	Anlamsal hata.

Örnek:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Bu hata değişkeni, geçerli kod yürütmesi sırasında hatalara neden olan deyimlerin toplam sayısını döndürür. Bu değişken kod yürütmesinin başlangıcında her zaman için 0 olarak sıfırlanır.

Söz Dizimi:

```
ScriptErrorCount
```

ScriptErrorList

Bu hata değişkeni, son kod yürütmesi sırasında oluşan tüm kod hatalarının birleştirilmiş listesini içerir. Her bir hata, satır beslemesiyle ayrılır.

Söz Dizimi:

```
ScriptErrorList
```

2 Kod ifadeleri

İfadeler hem **LOAD** deyimi hem de **SELECT** deyimi içinde kullanılabilir. Burada açıklanan söz dizimi ve fonksiyonlar **LOAD** deyimi için geçerlidir, ancak **SELECT** deyimi için geçerli değildir; çünkü SELECT deyimi Qlik Sense tarafından değil, ODBC sürücüsü tarafından yorumlanır. Bununla birlikte, çoğu ODBC sürücüsü genellikle aşağıda açıklanan fonksiyonlardan bazılarını yorumlayabilir.

İfadeler bir söz dizimi halinde bir araya getirilmiş fonksiyonlardan, alanlardan ve işleçlerden oluşur.

Qlik Sense kodundaki tüm ifadeler, bir sayı ve/veya bir dize (hangisi uygunsa) döndürür. Mantıksal fonksiyonlar ve işleçler False için 0 ve True için -1 döndürür. Sayıdan dizeye ve dizeden sayıya dönüştürmeler örtüktür. Mantıksal işleçler ve fonksiyonlar 0 değerini False ve diğer tüm değerleri True olarak yorumlar.

Bir ifade için genel söz dizimi:

Genel söz dizimi

İfade	Alanlar	İşleç
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

burada:

- **constant** tekli tırnak işareti içine alınmış bir dize (metin, tarih veya zaman) veya bir sayıdır. Sabitler, binlik ayırıcı olmadan ve ondalık ayırıcı olarak ondalık noktası ile yazılır.
- **fieldref**, yüklenen tablonun bir alan adıdır.
- **operator1**, (bir ifade üzerinde çalışan ve sağda yer alan) birli işleçtir.
- **operator2**, (iki ifade üzerinde çalışan ve her iki tarafta da birer tane olan) ikili işleçtir.
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Parametrelerin sayısı ve türleri rastgele değildir. Kullanılan fonksiyona bağlıdır.

Bu sayede ifadeler ve fonksiyonlar serbestçe iç içe geçebilir ve bir ifade yorumlanabilen bir değer döndürdüğü sürece Qlik Sense herhangi bir hata mesajı vermez.

3 Grafik ifadeleri

Bir grafik (görselleştirme) ifadesi fonksiyonların, alanların ve matematik işleçlerin (+ * / =) ve diğer hesaplamaların bir bileşimidir. İfadeler, görselleştirmede görülebilecek bir sonuç elde etmek amacıyla uygulamadaki verilerin işlenmesinde kullanılır. Kullanımları hesaplamalar ile sınırlı değildir. Başlıklar, alt başlıklar, dipnotlar ve hatta boyutlara yönelik ifadelerle daha dinamik ve güçlü görselleştirmeler oluşturabilirsiniz.

Bir başka deyişle, örneğin, bir görselleştirmenin başlığı statik metin olmak yerine, yapılan seçimlere göre sonucu değişen bir ifadeden oluşabilir.

Kod fonksiyonları ve grafik fonksiyonlarına ilişkin ayrıntılı referans için bkz. Kod söz dizimi ve grafik fonksiyonları.

3.1 Toplama kapsamını tanımlama

Genellikle, bir ifadede toplama değerini tanımlamak için kullanılan kayıtları birlikte belirleyen iki faktör vardır. Görselleştirmelerde çalışırken bu faktörler şunlardır:

- Boyutsal değer (bir grafik ifadesindeki toplama için)
- Seçimler

Bu faktörler birlikte, toplamanın kapsamını belirler. Hesaplamanızın seçimi, boyutu veya ikisini birden göz ardı etmesini isteyebileceğiniz durumlarla karşılaşabilirsiniz. Grafik fonksiyonlarında TOTAL niteleyicisini, set analizini veya ikisinin bir birleşimini kullanarak bunu yapabilirsiniz.

Toplama: Yöntem ve açıklama

Yöntem	Açıklama
TOTAL niteleyicisi	Toplama işlevinizin içinde total niteleyicisi kullanıldığında boyutsal değer göz ardı edilir. Toplama, tüm olası alan değerleri üzerinde yapılır. **TOTAL** niteleyicisinin ardından açılı ayraçlar içindeki bir veya daha fazla alan adından oluşan bir liste gelebilir. Bu alan adları grafik boyut değişkenlerinin bir alt kümesi olmalıdır. Bu durumda, hesaplama listelenenler dışındaki tüm grafik boyut değişkenlerini göz ardı ederek yapılır; yani listelenen boyut alanlarındaki alan değerlerinin her bir kombinasyonu için bir değer döndürülür. Ayrıca, geçerli anda grafikte bir boyut olmayan alanlar da listeye dahil edilebilir. Bu, boyut alanlarının sabit olmadığı grup boyutları durumunda kullanışlı olabilir. Gruptaki tüm değişkenlerin listelenmesi, detaya inme düzey değişikliği olduğunda fonksiyonun çalışmasına neden olur.
Set analizi	Toplamanızın içinde set analizi kullanıldığında seçim geçersiz kılınır. Toplama, boyutlar genelinde bölünmüş tüm değerler üzerinde yapılır.

Yöntem	Açıklama
TOTAL niteleyicisi ve set analizi	Toplamanızın içinde **TOTAL** niteleyicisi ve set analizi kullanıldığında seçim geçersiz kılınır ve boyutlar göz ardı edilir.

ALL niteleyicisi	Toplamanızın içinde **ALL** niteleyicisi kullanıldığında seçim ve boyutlar göz ardı edilir. Eşdeğeri {1} set analizi ifadesi ve **TOTAL** niteleyicisi ile elde edilebilir:
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

Örnek: TOTAL niteleyicisi

Aşağıdaki örnekte, göreceli bir paylaşımı hesaplamak için TOTAL niteleyicisinin nasıl kullanılabileceği gösterilmektedir. Q2 seçildiği varsayılırsa, TOTAL kullanıldığında boyutlar göz ardı edilerek tüm değerlerin toplamı hesaplanır.

Örnek: Total niteleyicisi

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%

*Sayıları yüzde olarak göstermek için yüzde değeri olarak göstermek istediğiniz hesaplamanın özellikler panelinde, **Number formatting** altında **Number** öğesini seçin ve **Formatting** seçeneğinden **Simple** öğesini ve % biçimlerinden birini seçin.*

Örnek: Set analizi

Aşağıdaki örnekte, herhangi bir seçimde bulunulmadan önce veri kümeleri arasında bir karşılaştırma yapmak için set analizinin nasıl kullanılabileceği gösterilmektedir. Q2 seçildiği varsayılırsa {1} adlı set tanımı ile set analizi kullanıldığında seçimler göz ardı edilerek, ancak boyutlar halinde bölünmüş olarak tüm değerlerin toplamı hesaplanır.

Örnek: Set analizi

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%

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
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%

Örnek: TOTAL niteleyicisi ve set analizi

Aşağıdaki örnekte, herhangi bir seçimde bulunulmadan önce ve tüm boyutlar genelinde veri kümeleri arasında bir karşılaştırma yapmak için set analizi ile TOTAL niteleyicisinin nasıl birleştirilebileceği gösterilmektedir. Q2 seçildiği varsayılırsa, {1} set tanımı ve TOTAL niteleyicisi ile set analizi kullanıldığında seçimler ve boyutlar göz ardı edilerek tüm değerlerin toplamı hesaplanır.

Örnek: TOTAL niteleyicisi ve set analizi

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%

Örneklerde kullanılan veriler:

```
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 Set analizi

Bir uygulamada bir seçim yaptığınızda, veride bir kayıt alt seti tanımlarsınız. `Sum()`, `Max()`, `Min()`, `Avg()` ve `Count()` gibi toplama işlevleri bu alt kümeye göre hesaplanır.

Başka bir deyişle, seçiminiz toplama işleminin kapsamını, hesaplamaların yapıldığı kayıt kümesini tanımlar.

Set analizi, geçerli seçim tarafından tanımlanan kayıt kümesinden farklı bir kapsam tanımlamak için bir yol sağlar. Bu yeni kapsam, alternatif bir seçim olarak da görülebilir.

Bu, geçerli seçimi belirli bir değerle, örneğin geçen yılın değeri veya küresel pazar payıyla karşılaştırmak istediğinizde yararlı olabilir.

Set ifadeleri

Set ifadeleri, toplama işlevlerinin içinde ve küme parantezleri arasına alınmış olarak kullanılır. Örneğin:

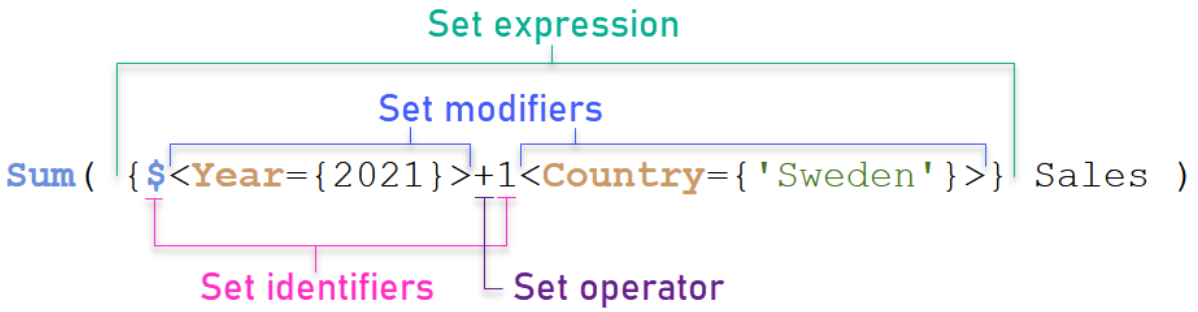
```
Sum( {$<Year={2021}>} Sales )
```

Set ifadesi aşağıdaki öğelerin birleşiminden oluşur:

- **Tanımlayıcılar**. Bir set tanımlayıcısı, başka bir yerde tanımlanan bir seçimi temsil eder. Ayrıca verilerdeki belirli bir kayıt setini temsil eder. Geçerli seçim, bir seçim iminden seçim veya alternatif bir durumdan seçim olabilir. Basit bir ifade tek bir tanımlayıcıdan (geçerli seçimdeki tüm kayıtlar anlamına gelen {$} işareti gibi) oluşur.
  Örnekler: `$`, `1`, `BookMark1`, `State2`
- **İşleçler**. Bir set işleci, farklı set tanımlayıcıları arasında birleşimler, farklılıklar veya kesişimler oluşturmak için kullanılabilir. Bu şekilde, set tanımlayıcıları tarafından tanımlanan seçimlerin bir alt kümesini veya bir üst kümesini oluşturabilirsiniz.
  Örnekler: `+`, `-`, `*`, `/`
- **Değiştiriciler**. Seçimini değiştirmek için set tanımlayıcısına bir set değiştirici eklenebilir. Değiştirici kendi başına da kullanılabilir, bu durumda varsayılan tanımlayıcıyı değiştirir. Değiştirici, köşeli parantez (<…>) içine alınmalıdır.
  Örnekler: `<Year={2020}>`, `<Supplier={ACME}>`

Öğeler birleştirilerek set ifadeleri oluşturulur.

Set ifadesindeki öğeler



Örneğin yukarıdaki set ifadesi `Sum(Sales)` toplamasından oluşturulmuştur.

İlk işlenen, geçerli seçim için `2021` yılının satışlarını döndürür. Bu durum, `$` set tanımlayıcısı ve `2021` yılının seçimini içeren değiştirici tarafından belirtilmektedir. İkinci işlenen, `Sweden` için `Sales` değerini döndürür ve `1` set tanımlayıcısı tarafından belirtilen geçerli seçimi yok sayar.

Son olarak söz konusu ifade, `+` set işlecinde belirtildiği üzere iki set işleneninden herhangi birine ait kayıtları içeren seti döndürür.

Örnekler

Yukarıdaki set ifadesi öğelerini birleştiren örnekler aşağıdaki konularda bulunabilir:

Doğal setler

Genellikle, bir set ifadesi hem veri modelindeki bir kayıt kümesini hem de bu veri alt kümesini tanımlayan bir seçimi temsil eder. Bu durumda sete doğal set denir.

Set tanımlayıcıları (set değiştiricileri olsun veya olmasın) her zaman doğal setleri temsil eder.

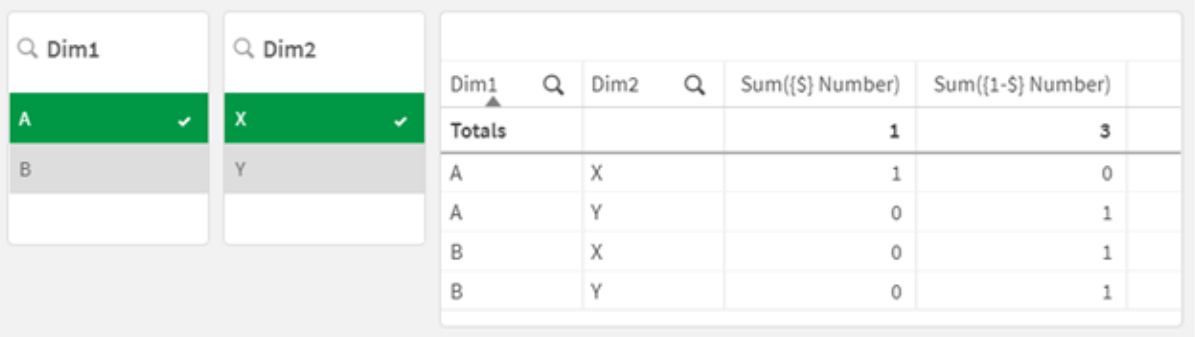
Bununla birlikte, set işleçlerini kullanan bir set ifadesi aynı zamanda kayıtların bir alt kümesini temsil eder, ancak yine de bir dizi alan değeri kullanılarak açıklanamaz. Böyle bir ifade doğal olmayan bir settir.

Örneğin, `{1-$}` tarafından verilen set her zaman bir seçimle tanımlanamaz. Bu nedenle doğal bir set değildir. Bu, aşağıdaki verileri yükleyerek, bir tabloya ekleyerek ve ardından filtre bölmelerini kullanıp seçimler yaparak gösterilebilir.

```
Load * Inline [Dim1, Dim2, Number A, X, 1 A, Y, 1 B, X, 1 B, Y, 1];
```

`Dim1` ve `Dim2` için seçimler yaparak aşağıdaki tabloda gösterilen görünümü elde edersiniz.

Doğal ve doğal olmayan setlerden oluşan tablo

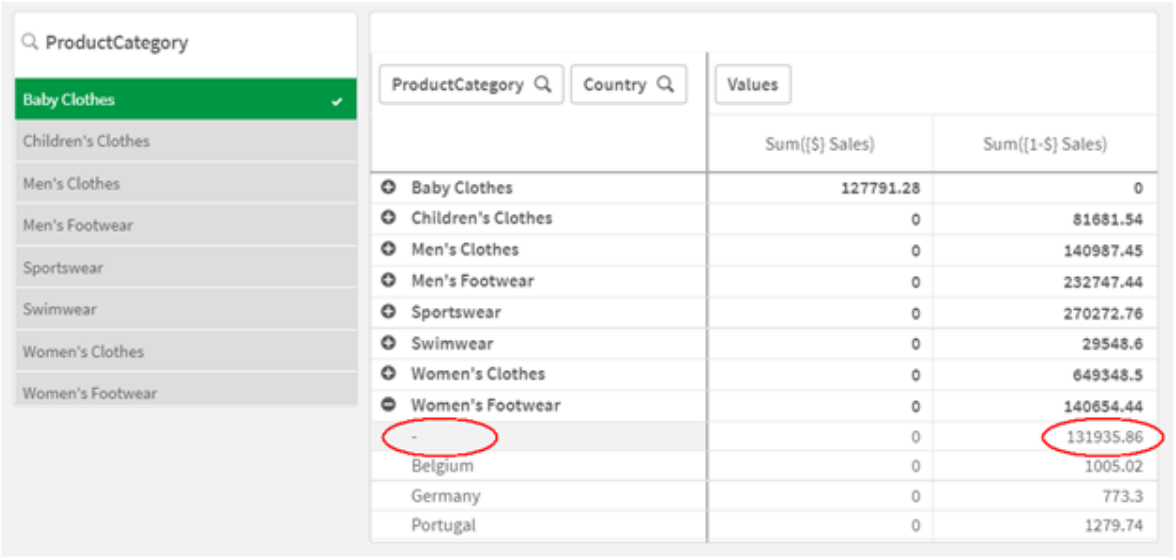


İlk hesaplamadaki set ifadesi doğal bir set kullanır: `{$}` yapılan seçime karşılık gelir.

İkinci hesaplama farklıdır. `{1-$}` kullanır. Bu sete karşılık gelen bir seçim yapmak mümkün değildir, dolayısıyla bu doğal olmayan bir settir.

Bu ayrımın birkaç sonucu vardır:

- Set değiştiricileri yalnızca set tanımlayıcılarına uygulanabilir. Rastgele bir set ifadesine uygulanamaz. Örneğin, aşağıdaki gibi bir set ifadesi kullanmak mümkün değildir:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Burada, normal (yuvarlak) parantezler, set değiştirici uygulanmadan önce `BM01` ile `BM02` arasındaki kesişimin değerlendirilmesi gerektiğini belirtir. Bunun nedeni, değiştirilebilecek bir öğe seti olmamasıdır.
- Doğal olmayan setleri `P()` ve `E()` öğe fonksiyonları içinde kullanamazsınız. Bu fonksiyonlar bir öğe seti döndürür, ancak öğe setini doğal olmayan bir setten çıkarmak mümkün değildir.
- Veri modelinde çok sayıda tablo varsa, doğal olmayan bir set kullanan bir hesaplama her zaman doğru boyutsal değerle ilişkilendirilemez. Örneğin aşağıdaki grafikte, hariç tutulan bazı satış rakamları doğru Country ile ilişkilendirilirken, diğerleri Country olarak NULL içerir.
  Doğal olmayan set içeren grafik



Atamanın doğru yapılıp yapılmadığı veri modeline bağlıdır. Bu durumda, seçim tarafından hariç tutulan bir ülkeye aitse numara atanamaz.

Set tanımlayıcıları

Bir set tanımlayıcısı, verilerdeki bir dizi kaydı (ya tüm verileri ya da verilerin bir alt kümesini) temsil eder. Bir seçim tarafından tanımlanan kayıt kümesidir. Geçerli seçim, tüm veriler (seçim yok), bir seçim iminden yapılan seçim veya alternatif bir durumdan bir seçim olabilir.

`Sum( {$<Year = {2009}>} Sales )` örneğinde, tanımlayıcı dolar işaretidir: `$`. Bu, geçerli seçimi temsil eder. Ayrıca tüm olası kayıtları temsil eder. Bu set daha sonra set ifadesinin değiştirici kısmı tarafından değiştirilebilir: `Year` içindeki `2009` seçim eklenir.

Daha karmaşık bir set ifadesinde, iki kayıt setinin bir birleşimini, farkını veya kesişimini oluşturmak için bir işleçle birlikte iki tanımlayıcı kullanılabilir.

Aşağıdaki tabloda bazı yaygın tanımlayıcılar gösterilmektedir.

Yaygın tanımlayıcıları içeren örnekler

Tanımlayıcı	Açıklama
`1`	Yapılan her tür seçimden bağımsız olarak, uygulamadaki tüm kayıtların tam kümesini temsil eder.
`$`	Varsayılan durumda geçerli seçimin kayıtlarını temsil eder. Bu nedenle `{$}` set ifadesi, genellikle bir set ifadesi belirtmemeye eşdeğerdir.
`$1`	Varsayılan durumda önceki seçimi temsil eder. `$2`, öncekinden bir önceki seçimi temsil eder ve bu böyle devam eder.

Tanımlayıcı	Açıklama
`$_1`	Sonraki (ileri) seçimi temsil eder. `$_2` sonrakinden bir sonraki seçimi temsil eder ve bu böyle devam eder.
`BM01`	Herhangi bir seçim imi kimliği veya seçim imi adı kullanabilirsiniz.
`AltState`	Durum adıyla, alternatif bir duruma başvurabilirsiniz.
`AltState::BM01`	Bir seçim imi tüm durumların seçimlerini içerir ve seçim imi adını belirterek belirli bir seçim imine başvurabilirsiniz.

Aşağıdaki tabloda farklı tanımlayıcılar içeren örnekler gösterilmektedir.

Farklı tanımlayıcılar içeren örnekler

Örnek	Sonuç
`Sum ({1} Sales)`	Seçimleri göz ardı ederek, ancak boyutu dikkate alarak, uygulama için toplam satışları döndürür.
`Sum ({$} Sales)`	Geçerli seçim için satışları döndürür; yani `Sum(Sales)` ile aynıdır.
`Sum ({$1} Sales)`	Önceki seçim için satışları döndürür.
`Sum ({BM01} Sales)`	`BM01` seçim imi adı için satışları döndürür.

Set işleçleri

Set işleçleri, veri setlerini dahil etmek, hariç tutmak veya kesiştirmek için kullanılır. Tüm işleçler, kümeleri işlenenler olarak kullanır ve sonuç olarak bir küme döndürür.

Set işleçlerini iki farklı durumda kullanabilirsiniz:

- Verilerdeki kayıt setlerini temsil eden, set tanımlayıcıları üzerinde bir set işlemi gerçekleştirmek için.
- Öğe setlerinde, alan değerlerinde veya bir set değiştirici içinde set işlemi gerçekleştirmek için.

Aşağıdaki tabloda, set ifadelerinde kullanılabilecek işleçler gösterilmektedir.

İşleçler

İşleç	Açıklama
`+`	Birleşim. Bu ikili işlem, iki küme işleneninden herhangi birine ait olan kayıtlardan veya öğelerden oluşan bir küme döndürür.
`-`	Hariç Tutma. Bu ikili işlem, iki küme işleneninden birincisine ait olan ancak diğerine ait olmayan kayıtlardan veya öğelerden oluşan bir küme döndürür. Ayrıca, bir birli işleç olarak kullanıldığında, tümleyen kümesini döndürür.
`*`	Kesişim. Bu ikili işlem, her iki küme işlenenine ait olan kayıtlardan öğelerden oluşan bir küme döndürür.
`/`	Simetrik fark `(XOR)`. Bu ikili işlem, her iki küme işlenenine ait olan kayıtlardan öğelerden oluşan bir küme döndürür.

Aşağıdaki tabloda işleçlerle ilgili örnekler gösterilmektedir.

İşleçler içeren örnekler

Örnek	Sonuç
`Sum ({1-$} Sales)`	Geçerli seçim tarafından hariç tutulan her şey için, satışları döndürür.
`Sum ({$*BM01} Sales)`	Seçim ile seçim imi `BM01` arasındaki kesişim için satışları döndürür.
`Sum ({-($+BM01)} Sales)`	Seçim ve seçim imi `BM01` ile hariç tutulan satışları döndürür.
`Sum ({$<Year= {2009}>+1<Country= {'Sweden'}>} Sales)`	Mevcut seçimlerle ilişkili 2009 yılı satışlarını döndürür ve tüm yıllar boyunca `Sweden` ülkesiyle ilişkili tüm veri setini ekler.
`Sum ({$<Country={"S*"}+ {"*land"}>} Sales)`	`S` ile başlayan veya `land` ile biten ülkeler için satışları döndürür.

Set değiştiricileri

Set ifadeleri bir hesaplamanın kapsamını tanımlamak için kullanılır. Set ifadesinin orta kısmı bir seçim belirten set değiştiricidir. Bu, kullanıcı seçimini veya set tanımlayıcısındaki seçimi değiştirmek için kullanılır ve sonuç, hesaplama için yeni bir kapsam tanımlar.

Küme değiştiricisi bir veya daha fazla alan adından oluşur ve her birinin ardından o alanda yapılması gereken bir seçim yapılır. Değiştirici, köşeli parantezler arasına alınır: `< >`

Örneğin:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Öğe setleri

Öğe seti, aşağıdakiler kullanılarak tanımlanabilir:

- Bir değer listesi
- Bir arama
- Başka bir alana başvuru
- Bir dizi fonksiyon

Öğe seti tanımı atlanırsa, set değiştirici bu alandaki herhangi bir seçimi temizleyecektir. Örneğin:

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

Örnekler: Öğe setlerine dayalı set değiştiriciler için grafik ifadeleri

Örnekler - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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 ];
```

Grafik ifadeleri

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

Tablo - Öğe setlerini temel alan set değiştiriciler

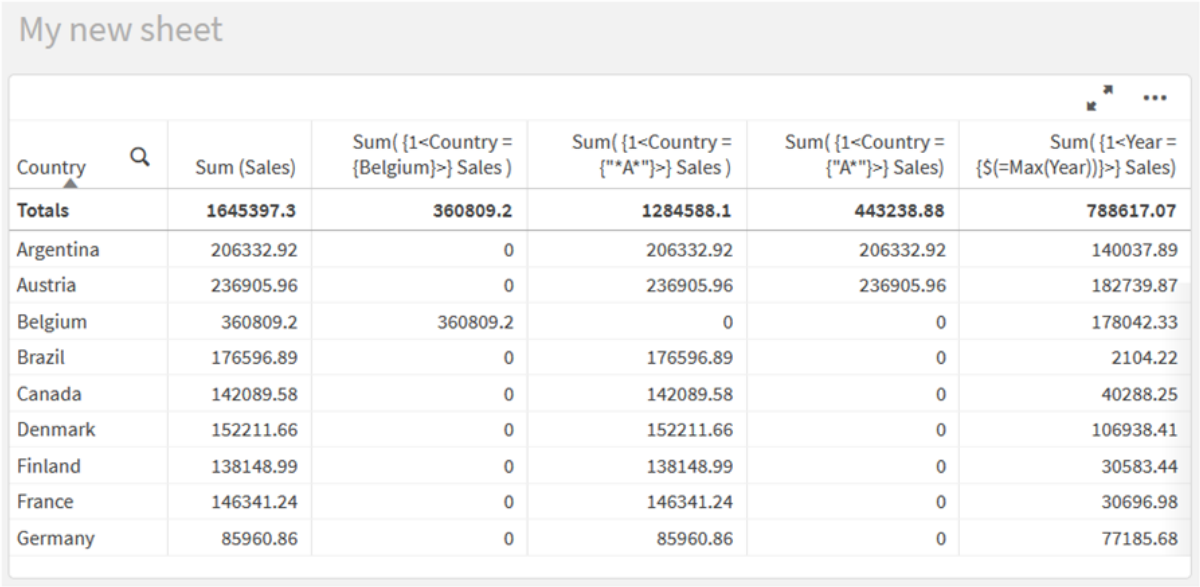
Ülke	Sum (Sales)	Sum({1<Country={Belgium}>}Sales)	Sum ({1<Country={"*A*"}>}Sales)	Sum ({1<Country={"A*"}>}Sales)	Sum({1<Year={$(=Max (Year))}>}Sales)
Toplamlar	1645397.3	360809.2	1284588.1	443238.88	788617.07
Arjantin	206332.92	0	206332.92	206332.92	140037.89
Avusturya	236905.96	0	236905.96	236905.96	182739.87
Belçika	360809.2	360809.2	0	0	178042.33
Brezilya	176596.89	0	176596.89	0	2104.22
Kanada	142089.58	0	142089.58	0	40288.25
Danimarka	152211.66	0	152211.66	0	106938.41
Finlandiya	138148.99	0	138148.99	0	30583.44
Fransa	146341.24	0	146341.24	0	30696.98
Almanya	85960.86	0	85960.86	0	77185.68

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Sales)`
    Set ifadesi olmadan `Sales` toplamını al.

- `Sum({1<Country={Belgium}>}Sales)`
  `Belgium` öğesini seç, sonra ilgili `Sales` toplamını al.
- `Sum({1<Country={"*A*"}>}Sales)`
  `A` içeren tüm ülkeleri seç, sonra ilgili `Sales` toplamını al.
- `Sum({1<Country={"A*"}>}Sales)`
  `A` ile başlayan tüm ülkeleri seç, sonra ilgili `Sales` toplamını al.
- `Sum({1<Year={$(=Max(Year))}>}Sales)`
  `2015` olan `Max(Year)` değerini hesapla, sonra ilgili `Sales` toplamını al.

Öğe kümelerine dayalı set değiştiriciler



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

Listelenen değerler

Öğe setine en yaygın örnek, küme parantezleri içine alınmış alan değerleri listesine dayalı olandır. Örneğin:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

İç küme ayraçları öğe setini tanımlar. Bireysel değerler virgülle ayrılır.

Alıntılar ve büyük/küçük harf duyarlılığı

Değerlerde boşluklar veya özel karakterler varsa, değerlerin tırnak içine alınması gerekir. Tek tırnaklar, tek bir alan değeriyle gerçek, büyük/küçük harfe duyarlı bir eşleşmeyi belirtir. Çift tırnak, bir veya birkaç alan değeriyle büyük/küçük harfe duyarlı olmayan bir eşleşmeyi ifade eder. Örneğin:

- `<Country = {'New Zealand'}>`
  Yalnızca `New Zealand` ile eşleşir.
- `<Country = {"New Zealand"}>`
  `New Zealand`, `NEW ZEALAND` ve `new zealand` ile eşleşir.

Tarihler tırnak işareti içine alınmalı ve söz konusu alanın tarih biçimi kullanılmalıdır. Örneğin:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Çift tırnaklar; köşeli parantezlerle veya vurgu işaretleriyle değiştirilebilir.

Aramalar

Öğe setleri aramalar yoluyla da oluşturulabilir. Örneğin:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Metin aramalarında joker karakterler kullanılabilir: Yıldız işareti (`*`) herhangi bir sayıda karakteri, soru işareti (`?`) ise tek bir karakteri temsil eder. İlişkisel işleçler, sayısal aramaları tanımlamak için kullanılabilir.

Aramalar için her zaman çift tırnak kullanmalısınız. Aramalar büyük/küçük harf duyarlıdır.

Dolar işareti genişletmeleri

Öğe setiniz içinde bir hesaplama kullanmak istiyorsanız dolar işareti genişletmeleri gerekir. Örneğin, yalnızca mümkün olan son yıla bakmak istiyorsanız, şunları kullanabilirsiniz:

```
<Year = {$(=Max(Year))}>
```

Diğer alanlarda seçilen değerler

Değiştiriciler, başka bir alanın seçilen değerlerine dayalı olabilir. Örneğin:

```
<OrderDate = DeliveryDate>
```

Bu değiştirici, seçilen değerleri `DeliveryDate` öğesinden alır ve bunları `OrderDate` öğesine bir seçim olarak uygular. Birkaç yüzü aşacak kadar çok sayıda tekil değer mevcutsa, bu işlem CPU'yu yoğun olarak kullanır ve bu işlemden kaçınılmalıdır.

Öğe seti fonksiyonları

Öğe seti ayrıca `P()` set fonksiyonlarına (olası değerler) ve `E()`, hariç tutulan değerlere dayalı olabilir.

Örneğin, `Cap` ürününün satıldığı ülkeleri seçmek istiyorsanız şunları kullanabilirsiniz:

```
<Country = P({1<Product={Cap}>} Country)>
```

Benzer şekilde, `Cap` ürününün satılmadığı ülkeleri seçmek isterseniz şunları kullanabilirsiniz:

```
<Country = E({1<Product={Cap}>} Country)>
```

Set değiştiriciler içeren aramalar

Set değiştiriciler ile yapılan aramalar aracılığıyla set öğeleri oluşturabilirsiniz.

Örneğin:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Aramalar her zaman çift tırnak veya eğik tek tırnak içine alınmalıdır. Gerçek dizelerin (tek tırnak) ve aramaların (çift tırnak) karışımı olan bir liste kullanabilirsiniz. Örneğin:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Metin aramaları

Metin aramalarında joker karakterler ve diğer semboller kullanılabilir:

- Yıldız işareti (*) herhangi bir sayıda karakteri temsil eder.
- Soru işareti (?) tek bir karakteri temsil eder.
- Şapka işareti (^) bir sözcüğün başlangıcını gösterir.

Örneğin:

- `<Country = {"C*", "*land"}>`
  `C` ile başlayan veya `land` ile biten tüm ülkeleri eşleştir.
- `<Country = {"*^z*"}>`
  Bu, `New Zealand` gibi `z` ile başlayan bir sözcük içeren tüm ülkeleri seçer.

Sayısal aramalar

Şu ilişkisel işleçleri kullanarak sayısal aramalar yapabilirsiniz: `>`, `>=`, `<`, `<=`

Sayısal arama, her zaman bu işleçlerden biriyle başlar. Örneğin:

- `<Year = {">2015"}>`
  2016 ve sonraki yılları eşleştir.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  2015'teki tüm tarihleri eşleştir. İki tarih arasındaki zaman aralığını betimlemek için kullanılan söz dizimine dikkat edin. Tarih biçiminin söz konusu alanın tarih biçimiyle eşleşmesi gerekir.

İfade aramaları

Daha gelişmiş aramalar yapmak için ifade aramalarını kullanabilirsiniz. Daha sonra, arama alanındaki her alan değeri için bir toplama değerlendirilir. Arama ifadesinin true sonucunu döndürdüğü tüm değerler seçilir.

İfade araması, her zaman bir eşittir işareti ile başlar: `=`

Örneğin:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Bu, satış değeri 1000'den büyük olan tüm müşterileri döndürür. `Sum(Sales)` mevcut seçimde hesaplanır. Bu, `Product` alanı gibi başka bir alanda bir seçiminiz varsa, yalnızca seçili ürünler için satış koşulunu karşılayan müşterilerin getirileceği anlamına gelir.

Koşulun seçimden bağımsız olmasını istiyorsanız, arama dizesi içinde set analizi kullanmanız gerekir. Örneğin:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Eşittir işaretinden sonraki ifadeler, boole değeri olarak yorumlanır. Bu, ifade başka bir değerle sonuçlanırsa, sıfır olmayan sayıların true, sıfırın ve dizelerin ise false olarak yorumlanacağı anlamına gelir.

Tırnak işaretleri

Arama dizeleri boş olduğunda veya özel karakterler içerdiğinde tırnak işaretleri kullanın. Tek tırnaklar, bir alan değeriyle gerçek, büyük/küçük harfe duyarlı bir eşleşmeyi belirtir. Çift tırnaklar, birden fazla alan değeriyle eşleşebilecek büyük/küçük harfe duyarsız bir aramayı belirtir.

Örneğin:

- `<Country = {'New Zealand'}>`
  Yalnızca `New Zealand` ile eşleştir.
- `<Country = {"New Zealand"}>`
  `New Zealand`, `NEW ZEALAND` ve `new zealand` ile eşleştir

Çift tırnaklar; köşeli parantezlerle veya vurgu işaretleriyle değiştirilebilir.

> *Qlik Sense hizmetinin önceki sürümlerinde, tek ve çift tırnak işaretleri arasında ayrım yoktu ve tırnak içine alınan tüm dizeler aynı şekilde aranıyordu. Geriye dönük uyumluluğu korumak için, Qlik Sense hizmetinin eski sürümleriyle oluşturulan uygulamalar, önceki sürümlerde olduğu gibi çalışmaya devam edecek. Qlik Sense Kasım 2017 veya sonrası ile oluşturulan uygulamalar iki tırnak türü arasındaki farkı tanır.*

Örnekler: Set değiştiriciler için grafik ifadeleri içeren aramalar

Örnekler - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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];
```

Örnek 1: Grafik ifadeler içeren metin aramaları

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

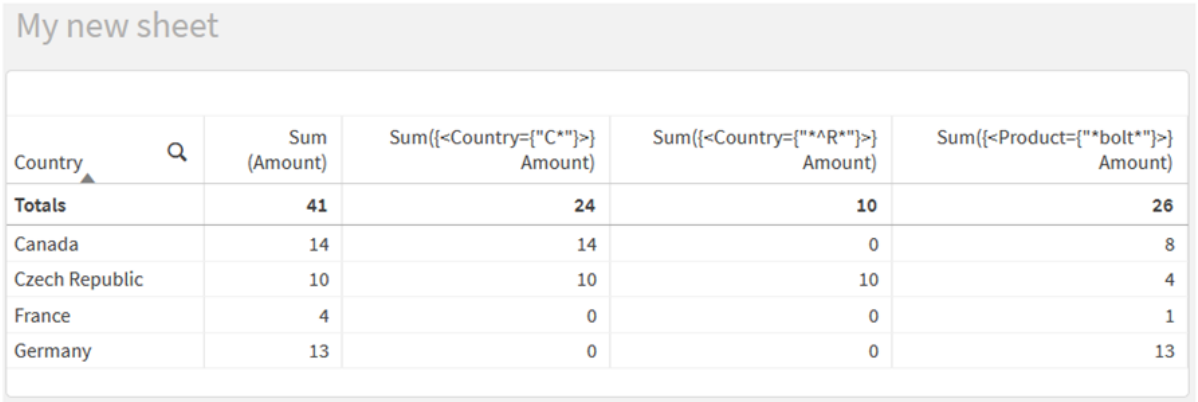
Tablo - Set değiştiriciler içeren metin aramaları

Ülke	Sum (Tutar)	Sum({<Country={"C*"}>} Tutar)	Sum({<Country={"*^R*"}>} Tutar)	Sum({<Product={"*bolt*"}>} Tutar)
Toplamlar	**41**	**24**	**10**	**26**
Kanada	14	14	0	8
Çek Cumhuriyeti	10	10	10	4
Fransa	4	0	0	1
Almanya	13	0	0	13

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Set ifadesi olmadan `Amount` toplamını al.
  - `Sum({<Country={"C*"}>}Amount)`
    `Canada` ve `Czech Republic` gibi `C` ile başlayan tüm ülkeler için `Amount` toplamını al.
  - `Sum({<Country={"*^R*"}>}Amount)`
    `Czech Republic` gibi `R` ile başlayan tüm ülkeler için `Amount` değerini topla.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    `Bolt` ve `Anchor bolt` gibi `bolt` dizesini içeren tüm ürünler için `Amount` değerini topla.

Set değiştiriciler içeren metin aramaları



Örnek 2: Grafik ifadeleri içeren sayısal aramalar

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

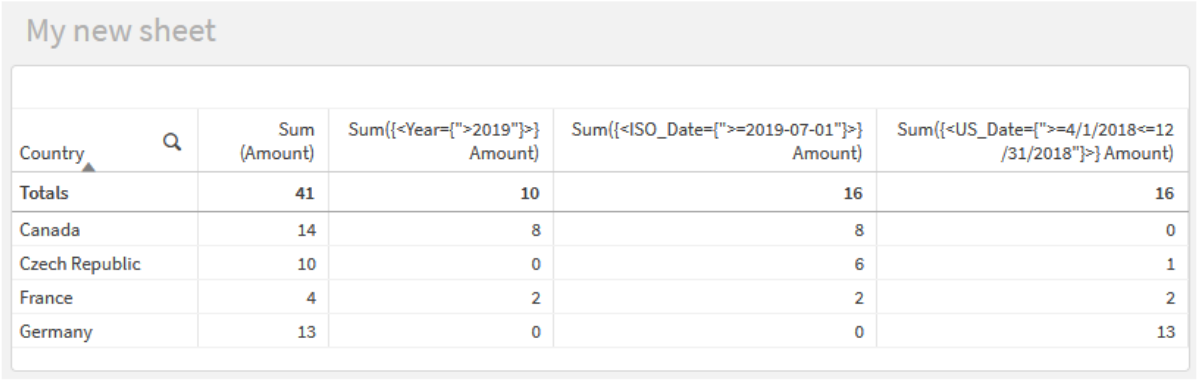
Tablo - Sayısal aramalar içeren set değiştiriciler

Ülke	Sum (Tutar)	Sum({<Year={">2019"}>} Tutar)	Sum({<ISO_Date={">=2019-07-01"}>} Tutar)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Tutar)
Toplamlar	**41**	**10**	**16**	**16**
Kanada	14	8	8	0
Çek Cumhuriyeti	10	0	6	1
Fransa	4	2	2	2
Almanya	13	0	0	13

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Set ifadesi olmadan `Amount` toplamını al.
  - `Sum({<Year={">2019"}>}Amount)`
    `2019` sonrası tüm yıllar için `Amount` değerini topla.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    `2019-07-01` veya sonraki tarihler için `Amount` değerlerini topla. Aramadaki tarihin biçimi, alanın biçimiyle eşleşmelidir.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Başlangıç ve bitiş tarihleri dahil `4/1/2018` ile `12/31/2018` arasındaki tüm tarihler için `Amount` değerlerini topla. Aramadaki tarihlerin biçimi, alanın biçimiyle eşleşmelidir.

Set değiştiriciler içeren sayısal aramalar



Örnek 3: Grafik ifadeleri içeren ifade aramaları

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

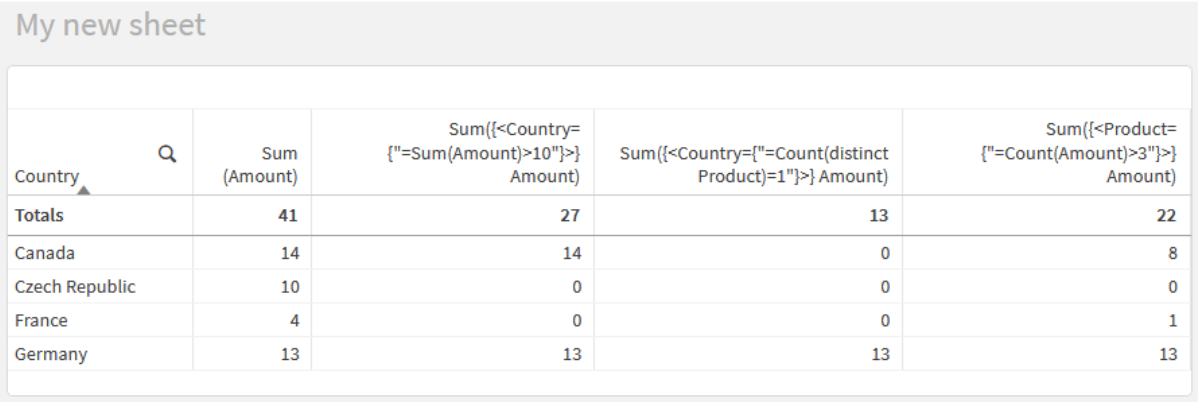
Table - Set modifiers with expression searches

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

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Set ifadesi olmadan `Amount` toplamını al.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Birleştirilmiş `Amount` toplam değeri `10` üzerinde olan tüm ülkeler için `Amount` değerini topla.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Tam olarak tek bir ürünle ilişkili tüm ülkeler için `Amount` değerini topla.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Verilerinde üçten fazla işlem olan tüm ülkeler için `Amount` değerini topla.

Set değiştiriciler içeren ifade aramaları



Dolar işareti genişletmeleri içeren set değiştiricileri

Dolar işareti genişletmeleri, ifade ayrıştırılıp değerlendirilmeden önce hesaplanan kurgulardır. Sonuç, daha sonra `$(...)` yerine ifadenin içine eklenir. İfade, daha sonra dolar işareti genişletmesinin sonucu kullanılarak hesaplanır.

İfade düzenleyicisi; dolar işareti genişletmesinin değerlendirme sonucunu doğrulayabilmeniz için, bir dolar işareti genişletmesi önizlemesi gösterir.

İfade düzenleyicisindeki dolar işareti genişletmesi önizlemesi

Dolar işareti genişletmelerini, öğe setinizin içinde bir hesaplama kullanmak istediğinizde kullanın.

Örneğin, yalnızca olabilecek en son yıla bakmak istiyorsanız, aşağıdaki kurguyu kullanabilirsiniz:

```
<Year = {$(=Max(Year))}>
```

Önce `Max(Year)` hesaplanır ve sonuç, `$(…)` yerine ifadenin içine eklenir.

Dolar işareti genişletmesinden sonra sonuç, aşağıdaki gibi bir ifade olacaktır:

```
<Year = {2021}>
```

Dolar işareti genişletmesinin içindeki ifade, mevcut seçim temel alınarak hesaplanır. Bu, başka bir alanda bir seçiminiz varsa, sonucun bundan etkileneceği anlamına gelir.

Hesaplamanın seçimden bağımsız olmasını istiyorsanız, dolar işareti genişletmesinin içinde set analizi kullanın. Örneğin:

```
<Year = {$(=Max({1} Year))}>
```

Dizeler

Dolar işareti genişletmesinin bir dize ile sonuçlanmasını istiyorsanız, normal tırnak işareti kuralları geçerlidir. Örneğin:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Dolar işareti genişletmesinden sonra sonuç, aşağıdaki gibi bir ifade olacaktır:

```
<Country = {'New Zealand'}>
```

Tırnak işaretleri kullanmazsanız, bir söz dizimi hatası alırsınız.

Sayılar

Dolar işareti genişletmesinin bir sayı ile sonuçlanmasını istiyorsanız, genişletmenin alan ile aynı biçimde olduğundan emin olun. Bu, ifadeyi bazen bir biçimlendirme fonksiyonu içine almanız gerektiği anlamına gelir.

Örneğin:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Dolar işareti genişletmesinden sonra sonuç, aşağıdaki gibi bir ifade olacaktır:

```
<Amount = {12362.00}>
```

Genişletmeyi her zaman ondalık basamak kullanmaya ve binler ayracı kullanmamaya zorlamak için bir diyez işareti kullanın. Örneğin:

```
<Amount = {$(#=Max(Amount))}>
```

Tarihler

Dolar işareti genişletmesinin bir tarih ile sonuçlanmasını istiyorsanız, genişletmenin doğru biçimde olduğundan emin olun. Bu, ifadeyi bazen bir biçimlendirme fonksiyonu içine almanız gerektiği anlamına gelir.

Örneğin:

```
<Date = {'$(=Date(Max(Date)))'}>
```

Dolar işareti genişletmesinden sonra sonuç, aşağıdaki gibi bir ifade olacaktır:

```
<Date = {'12/31/2015'}>
```

Dizelerde olduğu gibi doğru tırnak karakterlerini kullanmanız gerekir.

Yaygın kullanım örneği, hesaplamanın son ay (veya yıl) ile sınırlı olmasının istendiği durumdur. Bu durumda `AddMonths()` fonksiyonu ile birlikte bir sayısal arama kullanabilirsiniz.

Örneğin:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Dolar işareti genişletmesinden sonra sonuç, aşağıdaki gibi bir ifade olacaktır:

```
<Date = {">=9/31/2021"}>
```

Bu, geçen ay gerçekleşen tüm etkinlikleri seçer.

Örnek: Set değiştiriciler için grafik ifadeleri içeren dolar işareti genişletmeleri

Örnek - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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];
```

Grafik ifadeleri içeren dolar işareti genişletmeleri

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

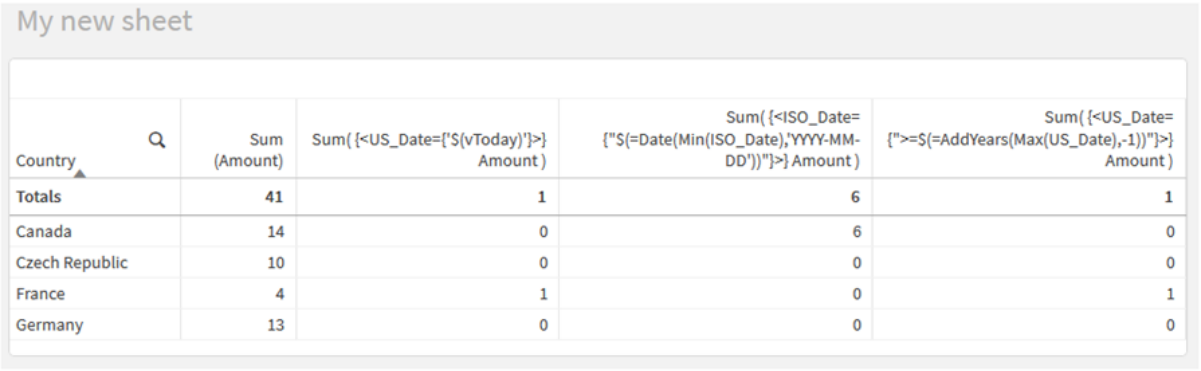
Tablo - Dolar işareti genişletmeleri içeren set değiştiriciler

Ülke	Sum (Tutar)	Sum({<US_Date= {'$(vToday)'}>} Tutar)	Sum({<ISO_Date= {"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Tutar)	Sum({<US_Date= {">=$(=AddYears(Max (US_Date),-1))"}>} Tutar)
Toplamlar	**41**	**1**	**6**	**1**
Kanada	14	0	6	0
Çek Cumhuriyeti	10	0	0	0
Fransa	4	1	0	1
Almanya	13	0	0	0

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Toplam Set ifadesi olmayan `Amount` değeri.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    `US_Date` değerinin `vToday` değişkenindeki gibi olduğu tüm kayıtlar için `Amount` değerini topla.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    `ISO_Date` değerinin olabilecek ilk (en küçük) `ISO_Date` ile aynı olduğu tüm kayıtlar için `Amount` değerini topla. `Date()` fonksiyonu, tarih biçiminin alanınkiyle eşleşmesini sağlamak için gereklidir.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Olabilecek en son (en büyük) `US_Date` tarihinden bir yıl önceki tarihte veya daha sonra bir `US_Date` içeren tüm kayıtlar için `Amount` değerini topla. `AddYears()` fonksiyonu, `DateFormat` değişkeni ile belirtilen biçimde bir tarih döndürür ve bunun `US_Date` alanı ile eşleşmesi gerekir.

Dolar işareti genişletmeleri içeren set değiştiricileri


My new sheet

Country	Sum (Amount)	Sum({<US_Date={'$(vToday)'}>} Amount)	Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Set işleçleri içeren set değiştiricileri

Set işleçleri, farklı öğe setlerini dahil etmek, hariç tutmak veya kesiştirmek için kullanılır. Öğe setlerini tanımlamak için farklı yöntemleri birleştirirler.

İşleçler, set tanımlayıcıları için kullanılanlarla aynıdır.

İşleçler

İşleç	Açıklama
+	Birleşim. Bu ikili işlem, iki küme işleneninden herhangi birine ait olan kayıtlardan veya öğelerden oluşan bir küme döndürür.
-	Hariç Tutma. Bu ikili işlem, iki küme işleneninden birincisine ait olan ancak diğerine ait olmayan kayıtlardan veya öğelerden oluşan bir küme döndürür. Ayrıca, bir birli işleç olarak kullanıldığında, tümleyen kümesini döndürür.
*	Kesişim. Bu ikili işlem, her iki küme işlenenine ait olan kayıtlardan öğelerden oluşan bir küme döndürür.
/	Simetrik fark (XOR). Bu ikili işlem, her iki küme işlenenine ait olan kayıtlardan öğelerden oluşan bir küme döndürür.

Örneğin, aşağıdaki iki değiştirici aynı alan değerleri setini tanımlar:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

İki ifade de `1997` değerini ve `20` ile başlayan yılları seçer. Diğer bir deyişle bu, iki koşulun bileşimidir.

Set işleçleri, daha karmaşık tanımlara da izin verir. Örneğin:

```
<Year = {1997, "20*"} - {2000}>
```

Bu ifade, yukarıdakilerle aynı yılları seçer, ancak buna ek olarak `2000` yılını hariç tutar.

.

Örnekler: Set işleçleri içeren set değiştiriciler için grafik ifadeleri

Örnekler - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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];
```

Grafik ifadeleri

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

Tablo - Set işleçleri içeren set değiştiriciler

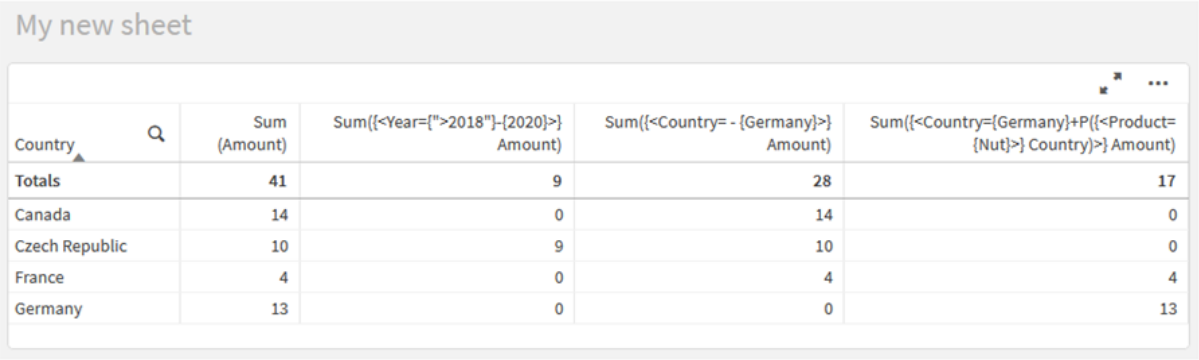
Ülke	$(=Sum (Tutar))	Sum({<Year={">2018"}-{2020}>} Tutar)	Sum ({<Country=-{Germany}>} Tutar)	Sum({<Country={Germany}+P({<Product={Nut}>}Ülke)>} Tutar)
Toplamlar	**41**	**9**	**28**	**17**
Kanada	14	0	14	0
Çek Cumhuriyeti	10	9	10	0
Fransa	4	0	4	4
Almanya	13	0	0	13

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Set ifadesi olmadan `Amount` toplamını al.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    `2020` dışında `2018` yılından sonraki tüm yıllar için `Amount` değerini topla.
  - `Sum({<Country=-{Germany}>}Amount)`
    `Germany` hariç tüm ülkeler için `Amount` değerini topla. Tekil dışlama işlecine dikkat edin.

- `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
  `Germany` ve `Nut` ürünü ile ilişkili tüm ülkeler için `Amount` değerini topla.

Set işleçleri içeren set değiştiricileri



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

Örtük set işleçleri içeren set değiştiriciler

Bir set değiştiricide seçim yazmanın standart yolu, eşittir işareti kullanmaktır. Örneğin:

```
Year = {">2015"}
```

Set değiştiricide eşittir işaretinin sağında kalan ifade, öğe seti olarak adlandırılır. Tek alan değerlerinin bir setini; yani bir seçimi tanımlar.

Bu gösterim, alandaki geçerli seçimi göz ardı ederek yeni bir seçim tanımlar. Bu nedenle set tanımlayıcısı bu alanda bir seçim içeriyorsa, eski seçim öğe setindekiyle değiştirilir.

Seçiminizde alandaki mevcut seçimi temel almak istediğinizde, farklı bir ifade kullanmanız gerekir

Örneğin, eski seçimi korumak ve yılın 2015'ten büyük olması koşulunu eklemek istiyorsanız, şunu yazabilirsiniz:

```
Year = Year * {">2015"}
```

Yıldız işareti, kesişimi tanımlayan bir set işlecidir, bu nedenle `Year` içindeki mevcut seçim ile yılın `2015` değerinden büyük olması koşulu arasındaki kesişimi elde edersiniz. Bu, şu şekilde de yazılabilir:

```
Year *= {">2015"}
```

Yani atama işleci (`*=`) örtük olarak bir kesişimi tanımlar.

Benzer şekilde örtük bileşimler, dışlamalar ve farklar da şunlar kullanılarak tanımlanabilir: `+=`, `-=`, `/=`

Örnekler: Örtük set işleçleri içeren set değiştiriciler için grafik ifadeleri

Örnekler - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
 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];
```

Örtük set işleçleri içeren grafik ifadeleri

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

Bir ülke listesinden `Canada` ve `Czech Republic` değerlerini seç.

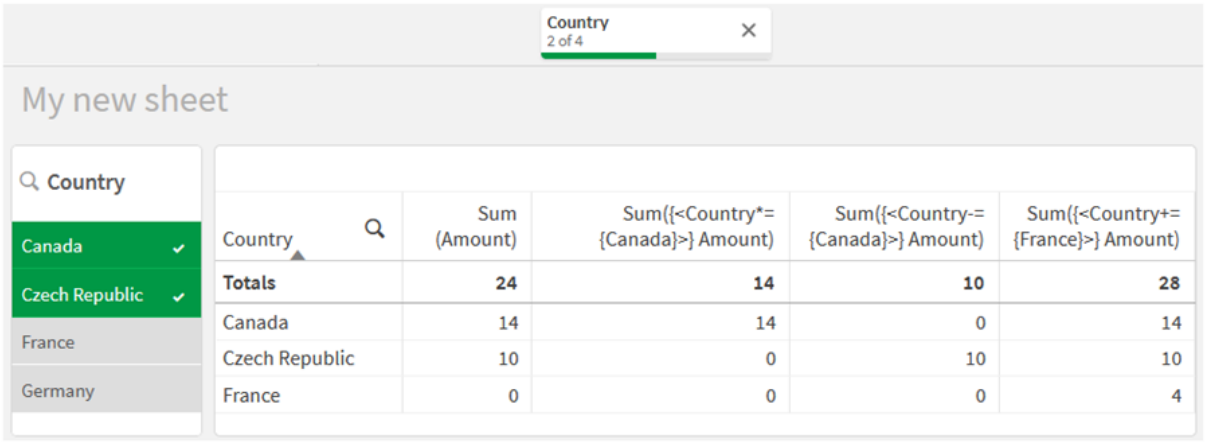
Tablo - Örtük set işleçleri içeren grafik ifadeleri

Ülke	$(=Sum (Tutar))	Sum({<Country*= {Canada}>} Tutar)	Sum({<Country-= {Canada}>} Tutar)	Sum({<Country+= {France}>} Tutar)
Toplamlar	**24**	**14**	**10**	**28**
Kanada	14	14	0	14
Çek Cumhuriyeti	10	0	10	10
Fransa	0	0	0	4

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Geçerli seçim için `Amount` değerini topla. Yalnızca `Canada` ve `Czech Republic` alanlarının sıfır olmayan değerler içerdiğine dikkat edin.
  - `Sum({<Country*={Canada}>}Amount)`
    Geçerli seçim için, `Country` değerinin `Canada` olması koşuluyla kesişimli olarak `Amount` toplamını al. `Canada`, kullanıcı seçiminin parçası değilse set ifadesi boş bir set döndürür ve sütundaki tüm satırlarda `0` değeri olur.
  - `Sum({<Country-={Canada}>}Amount)`
    Geçerli seçim için `Amount` değerini topla, ancak önce `Canada` değerini `Country` seçiminin dışında tut. `Canada` kullanıcı seçiminin parçası değilse, set ifadesi hiçbir sayıyı değiştirmez.
  - `Sum({<Country+={France}>}Amount)`
    Geçerli seçim için `Amount` değerini topla, ancak önce `France` değerini `Country` seçimine ekle. `France`, kullanıcı seçiminin zaten bir parçasıysa set ifadesi hiçbir bir sayıyı değiştirmez.

Örtük set işleçleri içeren set değiştiriciler



Set fonksiyonları kullanan set değiştiriciler

Bazen iç içe bir set tanımı kullanarak bir alan değerleri seti tanımlamanız gerekir. Örneğin, belirli bir ürünü satın almış olan tüm müşterileri, ürünü seçmeden seçmek isteyebilirsiniz.

Bu durumlarda, `P()` ve `E()` öğe seti işlevlerini kullanın. Bunlar sırasıyla bir alanda olabilecek değerleri ve alanın dışında tutulan değerleri döndürür. Köşeli parantezler içinde, söz konusu alanı ve kapsamı tanımlayan bir set ifadesini belirtebilirsiniz. Örneğin:

```
P({1<Year = {2021}>} Customer)
```

Bu, 2021'de işlem yapmış olan müşterilerin setini döndürür. Daha sonra bunu bir set değiştiricide kullanabilirsiniz. Örneğin:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Bu set ifadesi bu müşterileri seçer, ancak seçimi 2021 ile sınırlamaz.

Bu fonksiyonlar diğer ifadelerde kullanılamaz.

Ek olarak, öğe seti fonksiyonlarının içinde yalnızca doğal setler kullanılabilir. Doğal kümeden kasıt, basit bir seçimle tanımlanabilen bir kayıt kümesidir.

Örneğin, `{1-$}` ile verilen set her zaman bir seçim yoluyla tanımlanamaz ve bu nedenle doğal bir set değildir. Bu fonksiyonları doğal olmayan setlerde kullanmak beklenmeyen sonuçlar döndürür.

Örnekler: Set fonksiyonları kullanan set değiştiriciler için grafik ifadeleri

Örnekler - grafik ifadeleri

Yükleme kodu

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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];
```

Grafik ifadeleri

Aşağıdaki grafik ifadeleriyle Qlik Sense sayfasında bir tablo oluşturun.

Tablo - Set fonksiyonları kullanan set değiştiriciler

Ülke	$(=Sum (Tutar))	Sum({<Country=P ({<Year={2019}>} Ülke)>} Tutar)	Sum({<Product=P ({<Year={2019}>} Ürün)>} Tutar)	Sum({<Country=E ({<Product= {Washer}>} Ülke)>} Tutar)
Toplamlar	**41**	**10**	**17**	**13**
Kanada	14	0	6	0
Çek Cumhuriyeti	10	10	10	0
Fransa	4	0	1	0
Almanya	13	0	0	13

Açıklama

- Boyutlar:
  - `Country`
- Measures:
  - `Sum(Amount)`
    Set ifadesi olmadan `Amount` toplamını al.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    `2019` yılı ile ilişkili ülkeler için `Amount` değerini topla. Ancak hesaplama `2019` ile sınırlanmaz.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    `2019` ile ilişkili ürünler için `Amount` değerini topla. Ancak hesaplama `2019` ile sınırlanmaz.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    `Washer` ile ilişkili ülkeler için `Amount` değerini topla.

Set fonksiyonları kullanan set değiştiriciler

My new sheet

Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>} Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>} Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Öğretici - Bir küme ifadesi oluşturma

Veri analizini desteklemek için set ifadeleri oluşturabilirsiniz. Bu bağlamda analiz genellikle set analizi olarak adlandırılır. Set analizi, bir uygulamadaki mevcut seçimle tanımlanan kayıt setinden farklı bir kapsam tanımlamak için bir yol sunar.

Ne öğreneceksiniz?

Bu öğretici; set değiştiricilerini, tanımlayıcıları ve işleçleri kullanarak set ifadeleri oluşturmak için veri ve grafik ifadeleri sağlar.

Kimler bu eğitimi tamamlamalıdır?

Bu öğretici, kod düzenleyicisi ve grafik ifadeleri ile rahatça çalışabilen uygulama geliştiriciler içindir.

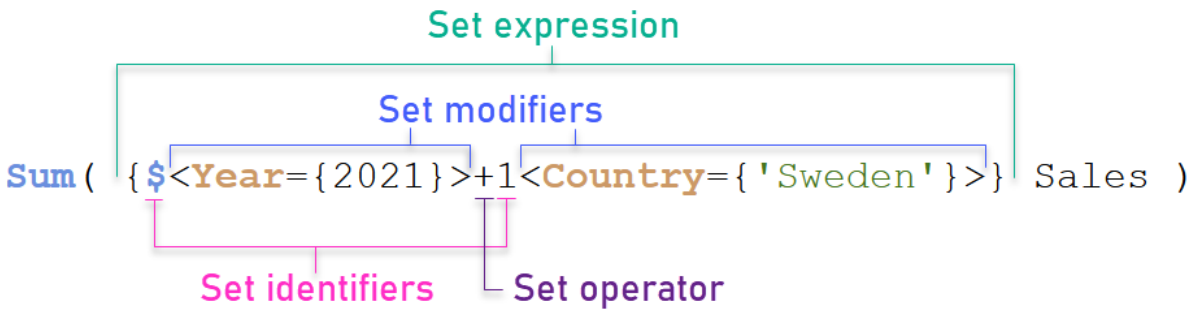
Başlamadan önce yapmanız gerekenler

Veri yüklemenize ve uygulama oluşturmanıza imkan tanıyan bir Qlik Sense Enterprise profesyonel erişim tahsisi.

Set ifadesindeki öğeler

Set ifadeleri `Sum()`, `Max()`, `Min()`, `Avg()` veya `Count()` gibi bir toplama fonksiyonu içine alınır. Set ifadeleri, öğeler olarak bilinen yapı taşlarından oluşturulur. Bu öğeler set değiştiriciler, tanımlayıcılar ve işleçlerdir.

Set ifadesindeki öğeler



Örneğin yukarıdaki set ifadesi `Sum(Sales)` toplamasından oluşturulmuştur. Set ifadesi küme ayraçları arasına alınır: `{ }`

İfadedeki ilk işlenen şudur: `$<Year={2021}>`

İşlenen, mevcut seçim için `2021` yılının satışlarını döndürür. `<Year={2021}>` değiştiricisi 2021 yılının seçimini içerir. `$` set tanımlayıcısı, set ifadesinin mevcut seçime dayandığını gösterir.

İfadedeki ikinci işlenen şudur: `1<Country={'Sweden'}>`

Bu işlenen, Sweden için Sales değerini döndürür. `<Country={'Sweden'}>` değiştiricisi, Sweden ülkesinin seçimini içerir. `1` set tanımlayıcısı, haritada yapılan seçimlerin yoksayılacağını gösterir.

Son olarak, `+` set işleci ifadenin iki set işleneninden herhangi birine ait olan kayıtlardan oluşan bir set döndüreceğini gösterir.

Set ifadesi oluşturma öğreticisi

Bu öğreticide gösterilen set ifadelerini oluşturmak için aşağıdaki prosedürleri tamamlayın.

Yeni bir uygulama oluşturun ve verileri yükleyin

Aşağıdakileri yapın:

1. Yeni bir uygulama oluşturun.
2. **Komut dosyası düzenleyicisi**'ne tıklayın. Alternatif olarak, gezinme çubuğundan **Hazırla** > **Veri yükleme düzenleyicisi**'ne tıklayın.
3. **Veri yükleme düzenleyicisi**'nde yeni bir bölüm oluşturun.
4. Aşağıdaki verileri kopyalayıp yeni bölüme yapıştırın: *Set ifadesi öğreticisi verileri (page 200)*
5. **Veri yükle**'ye tıklayın. Veri, satır için yükleme olarak yüklenir.

Değiştiricilerle set ifadeleri oluşturma

Küme değiştiricisi bir veya daha fazla alan adından oluşur ve her birinin ardından o alanda yapılması gereken bir seçim yapılır. Değiştirici, köşeli parantezler arasına alınır. Örneğin, bu set ifadesinde:

```
Sum ( {<Year = {2015}>} Sales )
```

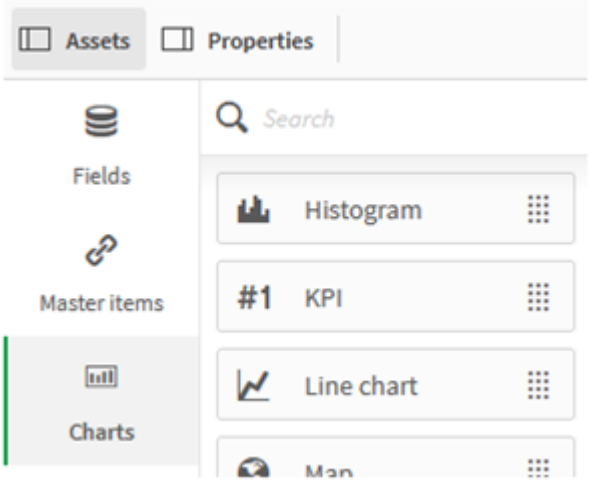
Değiştirici şudur:

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

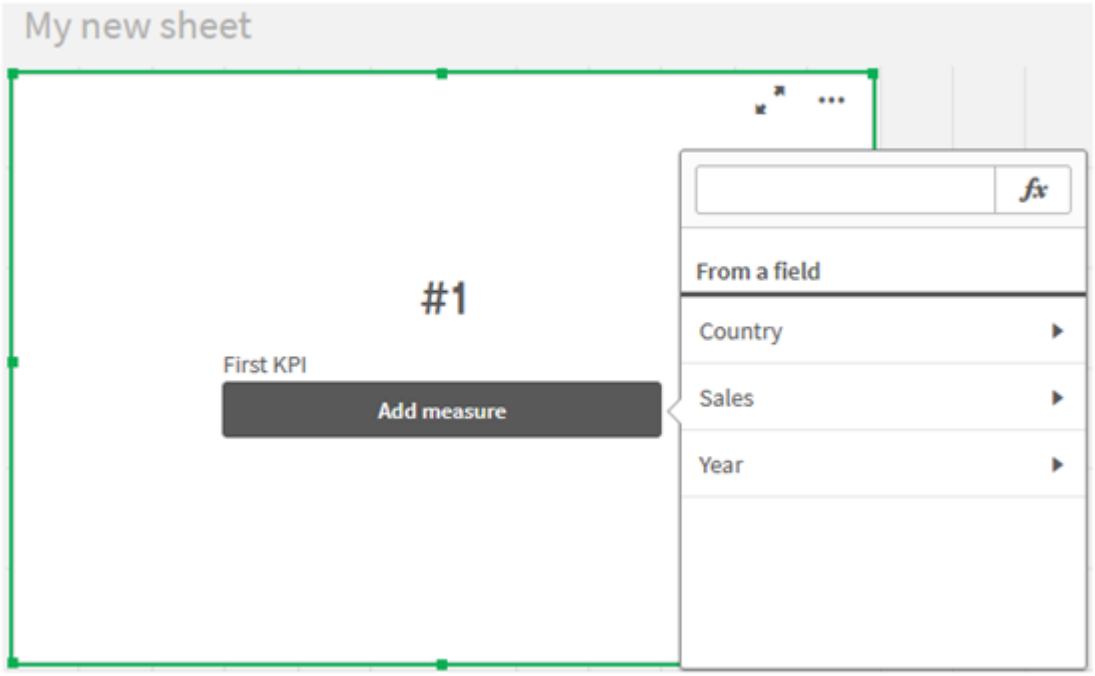
Bu değiştirici, 2015 yılından verilerin seçileceğini belirtir. Değiştiriciyi içine alan küme ayraçları bir set ifadesini gösterir.

Aşağıdakileri yapın:

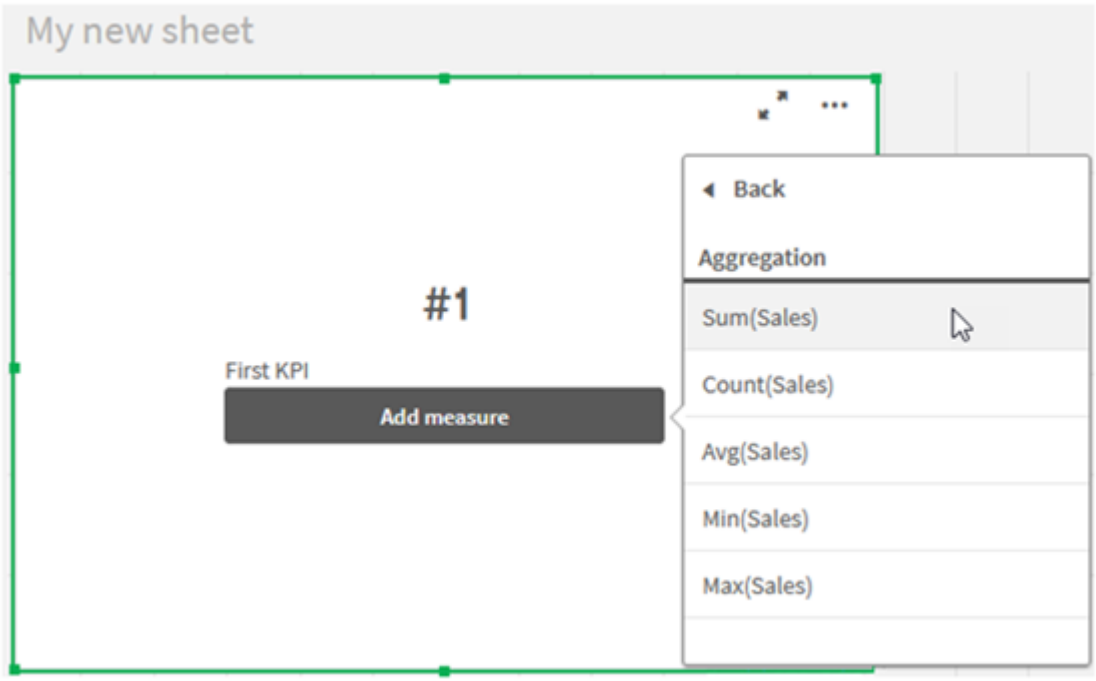
1. Bir sayfada, gezinti çubuğundan **Varlıklar** panelini açın, sonra **Grafikler**'e tıklayın.



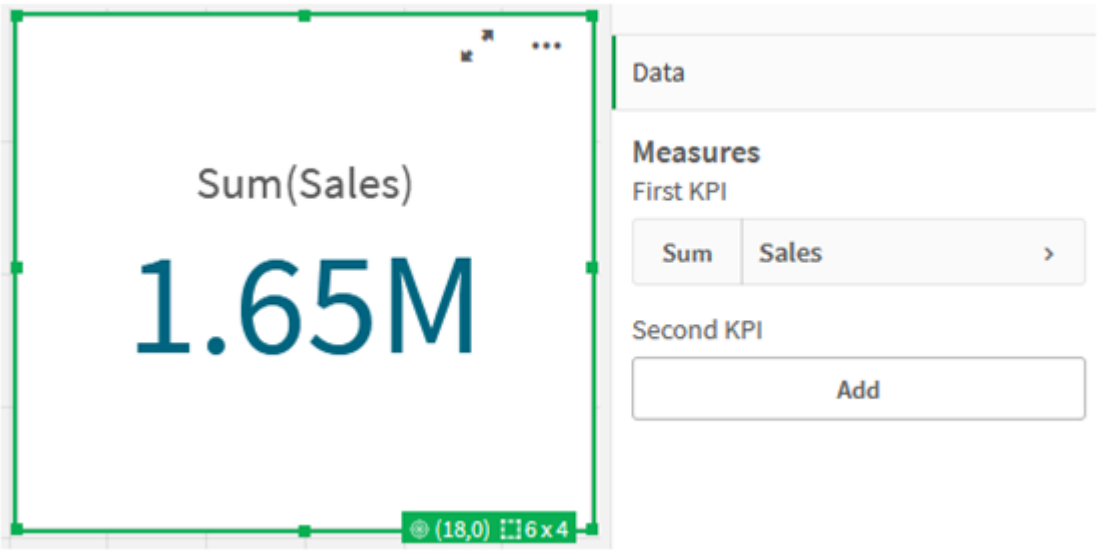
2. Bir **KPI**'yı sayfaya sürükleyin, sonra **Hesaplama ekle**'ye tıklayın.



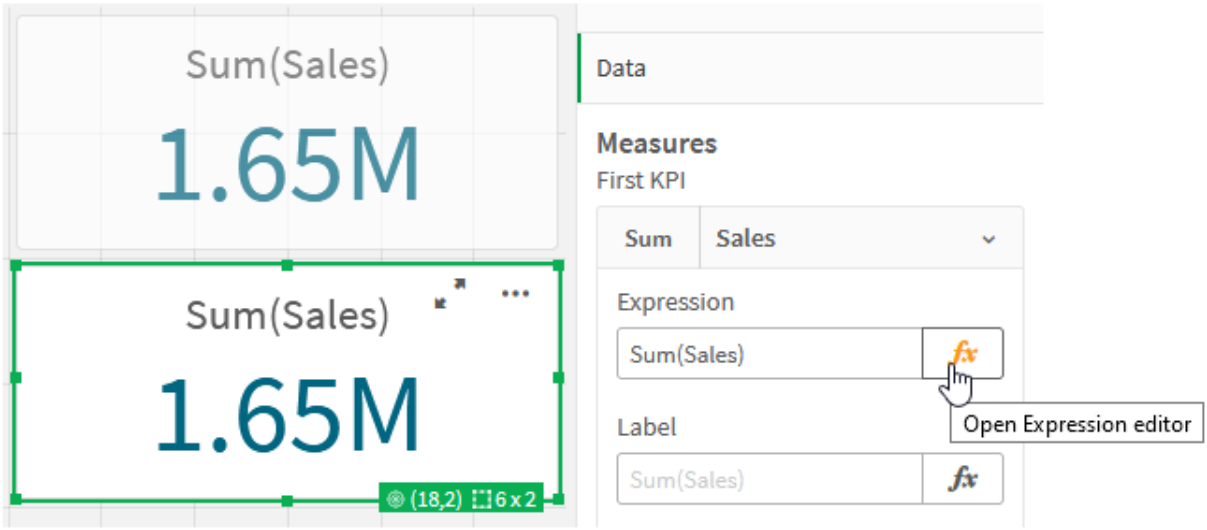
3. `Sales` üzerine tıklayın, sonra toplama için `Sum(Sales)` işlevini seçin.



KPI, tüm yıllar için satışların toplamını gösterir.



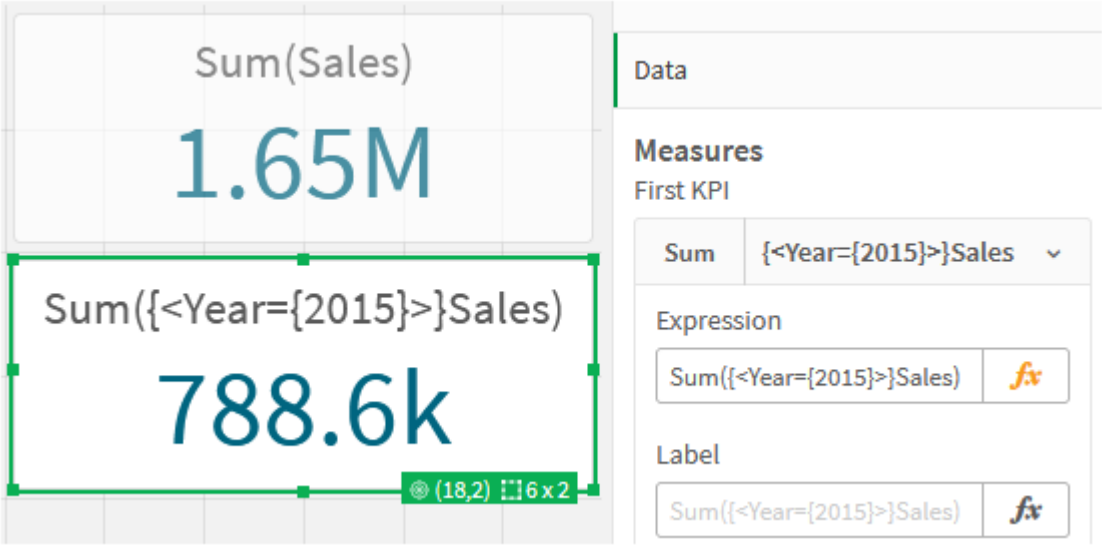
4. Yeni bir KPI oluşturmak için KPI ifadesini kopyalayıp yapıştırın.
5. Yeni KPI üzerine tıklayın, **Hesaplamalar**'ın altından **Sales**'e tıklayın, sonra **İfade düzenleyicisini aç**'a tıklayın.



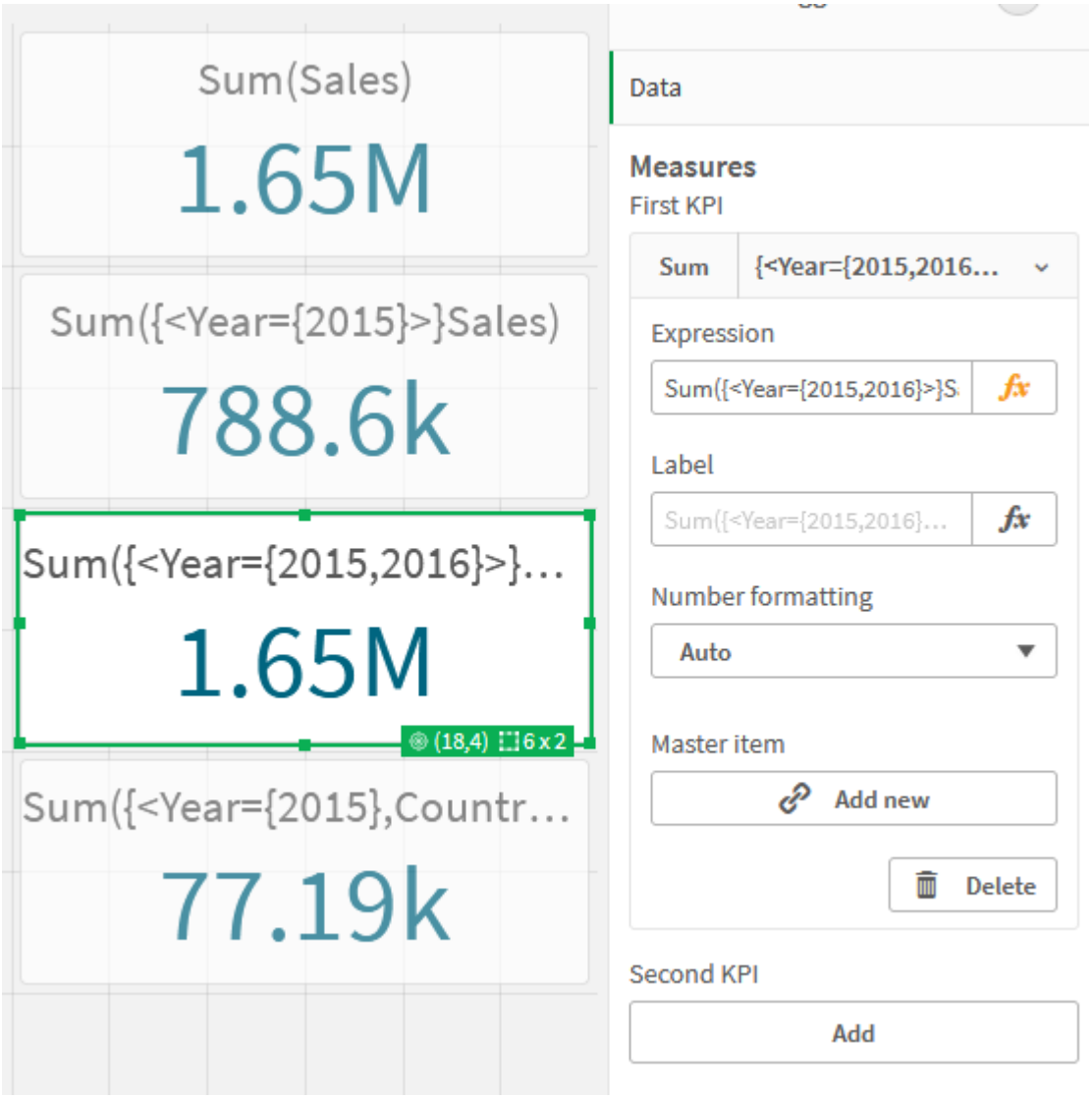
İfade düzenleyicisi `Sum(Sales)` toplaması ile açılır.

6. İfade düzenleyicisinde yalnızca 2015 için Sales değerlerini toplayacak bir ifade oluşturun:
   i. Bir set ifadesini belirtmek için küme ayraçları ekleyin: `Sum({}Sales)`
   i. Bir set değiştiriciyi göstermek için köşeli ayraç kullanın: `Sum({<>}Sales)`
   ii. Köşeli ayraçların arasına seçilecek alanı; burada `Year` alanının ardından eşittir işareti ekleyin. Sonra, 2015'i başka bir çift küme parantezinin arasına alın. Ortaya çıkan set değiştiricisi şudur: `{<Year={2015}>}`.
   İfadenin tamamı şudur:
   `Sum({<Year={2015}>}Sales)`

iii. İfadeyi kaydetmek ve ifade düzenleyicisini kapatmak için **Uygula**'ya tıklayın. Sales değerlerinin 2015 yılı için toplamı KPI içinde gösterilir.



7. Şu ifade ile iki veya daha fazla KPI ekleyin:
   `Sum({<Year={2015,2016}>}Sales)`
   Yukarıdaki değiştirici şudur: `<Year={2015,2016}>`. İfade, 2015 ve 2016 için Sales toplamını döndürür.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Yukarıdaki değiştirici şudur: `<Year={2015}, Country={'Germany'}>`. İfade, 2015 için Sales değerlerinin toplamını döndürür; burada 2015 Germany ile kesişmektedir.
   Set değiştirici kullanan KPI'lar



Set tanımlayıcılar ekleme

Yukarıdaki set ifadelerinde, tanımlayıcı kullanılmadığı için mevcut seçimlere uyulur. Sonra, seçimler yapıldığında davranışı belirtmek için tanımlayıcılar ekleyin.

Aşağıdakileri yapın:

Sayfanızda şu ifadeleri oluşturun veya sayfaya kopyalayın:

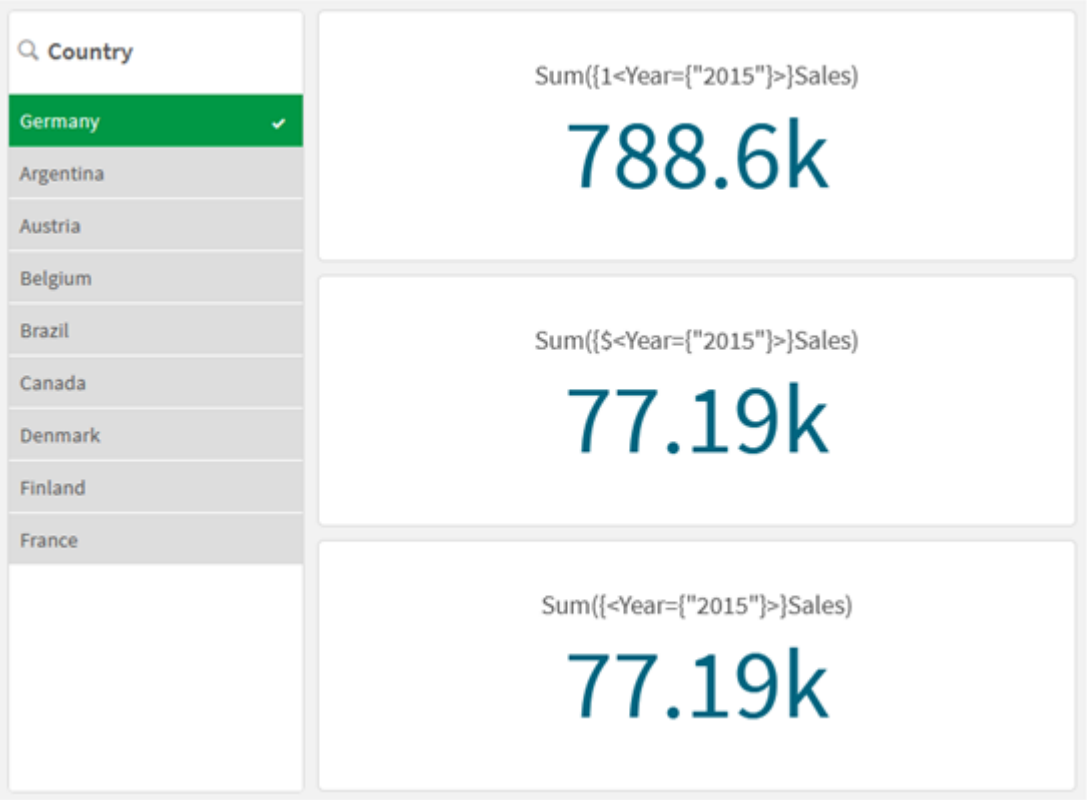
```
Sum({$<Year={"2015"}>}Sales
```

Set ifadesinde `$` tanımlayıcısı için veride yapılan mevcut seçimlere uyulur. Bu, ayrıca, bir tanımlayıcı kullanılmadığında varsayılan davranıştır.

```
Sum({1<Year={"2015"}>}Sales)
```

`1` tanımlayıcısı, 2015 üzerindeki `Sum(Sales)` toplamasının mevcut seçimi yok saymasına neden olur. Toplamanın değeri, kullanıcı başka seçimler yaptığında değişmez. Örneğin, aşağıda Germany seçildiğinde, 2015'in toplaması için toplam değişmez.

Set değiştiriciler ve tanımlayıcılar kullanan KPI'lar



İşleç ekleme

Set işleçleri, veri setlerini dahil etmek, hariç tutmak veya kesiştirmek için kullanılır. Tüm işleçler, kümeleri işlenenler olarak kullanır ve sonuç olarak bir küme döndürür.

Set işleçlerini iki farklı durumda kullanabilirsiniz:

- Verilerdeki kayıt setlerini temsil eden, set tanımlayıcıları üzerinde bir set işlemi gerçekleştirmek için.
- Öğe setlerinde, alan değerlerinde veya bir set değiştirici içinde set işlemi gerçekleştirmek için.

Aşağıdakileri yapın:

Sayfanızda şu set ifadesini oluşturun veya sayfaya kopyalayın:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

Artı işareti `(+)` işleci, 2015 ve Germany için veri setlerinin bileşimini üretir. Yukarıda set tanımlayıcılarında açıklandığı gibi, dolar işareti `($)` tanımlayıcısı, ilk işlenen (`<Year={2015}>`) için mevcut seçimlerin kullanılacağı anlamına gelir. `1` tanımlayıcısı, ikinci işlenen (`<Country={'Germany'}>`) için seçimin yoksayılacağı anlamına gelir.

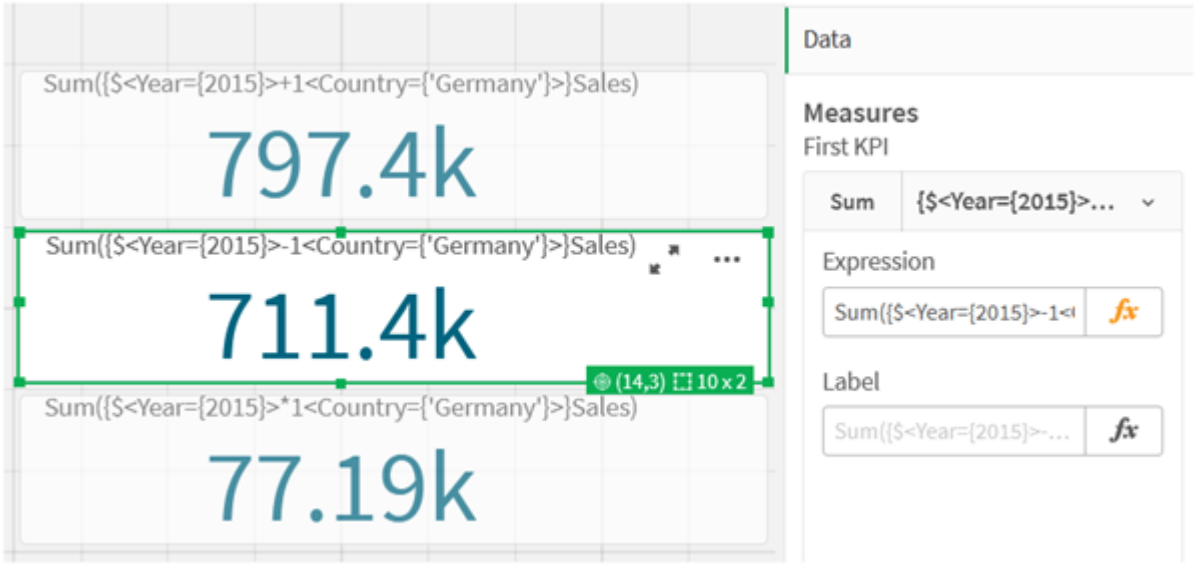
Artı işareti (+) işlecini kullanan KPI

Alternatif olarak, 2015 yılına ait olup Germany yılına ait olmayan kayıtlardan oluşan bir veri seti döndürmek için eksi işareti `(-)` kullanın. Veya her iki sete de ait olan kayıtlardan oluşan bir set döndürmek için bir yıldız `(*)` işareti kullanın.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

İşleç kullanan KPI'lar



Set ifadesi öğreticisi verileri

Komut dosyası

Aşağıdaki verileri satır içi yükleme olarak yükleyin, sonra öğreticideki grafik ifadelerini oluşturun.

```
//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 ];
```

Set ifadeleri için sözdizimi

Tam söz dizimi (önceliği tanımlamak üzere standart ayraçların isteğe bağlı kullanımını içermez) Backus-Naur Biçimciliği kullanılarak açıklanır:

```
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 Grafik ifadeleri için genel söz dizimi

Aşağıdaki genel söz dizimi yapısı, birçok isteğe bağlı parametre ile grafik ifadeleri için kullanılabilir:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

burada:

constant tekli tırnak işareti içine alınmış bir dize (metin, tarih veya zaman) veya bir sayıdır. Bunlar, binlik ayırıcı olmadan ve ondalık ayırıcı olarak da ondalık noktası ile yazılır.

expressionname, aynı grafikteki başka bir ifadenin adıdır (etikettir).

operator1, (bir ifade üzerinde çalışan ve sağda yer alan) birli işleçtir.

operator2, (iki ifade üzerinde çalışan ve her iki tarafta da birer tane olan) ikili işleçtir.

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```

Parametrelerin sayısı ve türleri rastgele değildir. Kullanılan fonksiyona bağlıdır.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```

Parametrelerin sayısı ve türleri rastgele değildir. Kullanılan fonksiyona bağlıdır.

3.4 Toplamalar için genel söz dizimi

Aşağıdaki genel söz dizimi yapısı, birçok isteğe bağlı parametre ile toplamalar için kullanılabilir:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```

fieldref bir alan adıdır.

```
functionaggr ::= functionname ( parameters2 )
```

Böylece, ifadeler ve fonksiyonlar serbestçe iç içe yerleştirilebilir; **fieldref** her zaman tam bir toplama işleviyle kapatıldığı sürece ve ifadenin yorumlanabilir bir değer döndürmesi şartıyla, Qlik Sense herhangi bir hata mesajı vermez.

4 İşleçler

Bu bölümde, Qlik Sense uygulamasında kullanılabilen işleçler açıklanmaktadır. İki tür işleç vardır:

- Birli işleçler (sadece bir işlenen alır)
- İkili işleçler (iki işlenen alır)

İşleçlerin çoğu ikili işleçtir.

Aşağıdaki işleçler tanımlanabilir.

- Bit işleçleri
- Mantıksal işleçler
- Sayısal işleçler
- İlişkisel işleçler
- Dize işleçleri

4.1 Bit işleçleri

Tüm bit işleçleri, işlenenleri işaretli tamsayılara (32 bit) dönüştürür (keser) ve sonucu aynı şekilde döndürür. Tüm işlemler bit'lerin tek tek işlenmesiyle gerçekleştirilir. İşlenen sayı olarak yorumlanamazsa, işlem NULL döndürür.

Bit işleçleri

İşleç	Adı soyadı	Açıklama
bitnot	Bit tersi.	Birli işleç. İşlem, teker teker gerçekleştirilen bit'leri işlenenin mantıksal tersini verir. **Örnek:** `bitnot 17`-18 döndürür
bitand	Bit ve.	İşlem, teker teker gerçekleştirilen bit'leri işlenenlerin mantıksal AND karşılığını verir. **Örnek:** `17 bitand 7` 1 döndürür
bitor	Bit veya.	İşlem, teker teker gerçekleştirilen bit'leri işlenenlerin mantıksal OR karşılığını verir. **Örnek:** `17 bitor 7` 23 döndürür

İşleç	Adı soyadı	Açıklama
bitxor	Bit özel veya.	İşlem, teker teker gerçekleştirilen bit'leri işlenenlerin mantıksal özel or karşılığını verir. **Örnek:** `17 bitxor 7` 22 döndürür
>>	Bit sağa kaydırma.	İşlem, ilk işleneni sağa kaydırılmış olarak döndürür. Adım sayısı ikinci işlenende tanımlanır. **Örnek:** `8 >> 2` 2 döndürür
<<	Bit sola kaydırma.	İşlem, ilk işleneni sola kaydırılmış olarak döndürür. Adım sayısı ikinci işlenende tanımlanır. **Örnek:** `8 << 2` 32 döndürür

4.2 Mantıksal işleçler

Tüm mantıksal işleçler, işlenenleri mantıksal olarak yorumlar ve sonuç olarak True (-1) veya False (0) döndürür.

Mantıksal işleçler

İşleç	Açıklama
not	Mantıksal ters sonuç. Az sayıdaki birli işleçlerden biridir. İşlem, işlenenin mantıksal tersini döndürür.
and	Mantıksal ve. İşlem, işlenenin mantıksal ve sonucunu döndürür.
or	Mantıksal veya. İşlem, işlenenin mantıksal veya sonucunu döndürür.
Xor	Mantıksal dışlamalı veya. İşlem, işlenenin mantıksal dışlamalı veya sonucunu döndürür. Diğer bir deyişle 'mantıksal veya' gibidir, ancak her iki işlenen True ise sonuç False olur.

4.3 Sayısal işleçler

Tüm sayısal işleçler, işlenenlerin sayısal değerlerini kullanır ve sonuç olarak bir sayısal değer döndürür.

Sayısal işleçler

İşleç	Açıklama
+	Pozitif sayı (birli işleç) veya aritmetik toplama işlemi işareti. İkili işlem, iki işlenenin toplamını döndürür.
-	Negatif sayı (birli işleç) veya aritmetik çıkarma işlemi işareti. Birli işlem, işlenenin -1 ile çarpımını ve ikili işlem iki işlenen arasında farkını döndürür.
*	Aritmetik çarpma işlemi. İşlem iki işlenenin ürününü döndürür.
/	Aritmetik bölme işlemi. İşlem iki işlenen arasındaki oranı döndürür.

4.4 İlişkisel işleçler

Tüm ilişkisel işleçler, işlenenlerin değerlerini karşılaştırır ve sonuç olarak True (-1) veya False (0) sonucunu döndürür. Tüm ilişkisel işleçler ikilidir.

İlişkisel işleçler

İşleç	Açıklama
<	Küçüktür. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.
<=	Küçüktür veya eşittir. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.
>	Büyüktür. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.
>=	Büyüktür veya eşittir. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.
=	Eşittir. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.
<>	Eşit değildir. Her iki işlenen sayısal olarak yorumlanabiliyorsa, sayısal bir karşılaştırma yapılır. İşlem, karşılaştırmanın değerlendirmesinin mantıksal değerini döndürür.

İşleç	Açıklama
precedes	**<** işlecinin aksine, karşılaştırma öncesinde bağımsız değişken değerlerinin sayısal yorumlamasını yapmaya yönelik bir girişimde bulunulmaz. İşlecin solundaki değerin, dize karşılaştırmasında sağdaki değerin metin temsilinden önce gelen bir metin temsiline sahip olması durumunda işlem true sonucunu döndürür. **Örnek:** `'1 ' precedes ' 2'` şunu döndürür: FALSE `' 1' precedes ' 2'` şunu döndürür: TRUE bir boşluğun (' ') ASCII değeri sayının ASCII değerinden az olduğundan. Bunu şununla karşılaştırın: `'1 ' < ' 2'` , TRUE değerini döndürür `' 1' < ' 2'` şunu döndürür: TRUE
follows	**>** işlecinin aksine, karşılaştırma öncesinde bağımsız değişken değerlerinin sayısal yorumlamasını yapmaya yönelik bir girişimde bulunulmaz. İşlecin solundaki değerin, dize karşılaştırmasında sağdaki değerin metin temsilinden sonra gelen bir metin temsiline sahip olması durumunda işlem true sonucunu döndürür. **Örnek:** `' 2' follows '1'`, FALSE değerini döndürür ' 2' follows ' 1' şunu döndürür: TRUE bir boşluğun (' ') ASCII değeri sayının ASCII değerinden az olduğundan. Bunu şununla karşılaştırın: `' 2' > ' 1'` şunu döndürür: TRUE `' 2' > '1 '` şunu döndürür: TRUE

4.5 Dize işleçleri

İki dize işleci mevcuttur. Bunlardan biri, işlenenlerin dize değerlerini kullanır ve sonuç olarak bir dize döndürür. Diğeri, işlenenleri karşılaştırır ve eşleşmeyi göstermek için bir boole değeri döndürür.

&

Dize birleşimi. İşlem, birbirini izleyen iki işlenen dizesinden oluşan bir metin dizesi döndürür.

Örnek:

`'abc' & 'xyz'`, 'abcxyz' döndürür.

like

Joker karakterlerle dize karşılaştırması. İşleçten önceki dize işleçten sonraki dizeyle eşleşirse, işlem, boole True (-1) sonucunu döndürür. İkinci dize, * (herhangi bir sayıda rastgele karakter) veya ? (bir rastgele karakter) joker karakterlerini içerebilir.

Örnek:

`'abc' like 'a*'` şunu döndürür: True (-1)

`'abcd' like 'a?c*'` şunu döndürür: True (-1)

`'abc' like 'a??bc'` şunu döndürür: False (0)

5 Kod ve grafik fonksiyonları

Veri yükleme kodlarında ve grafik ifadelerinde fonksiyonlar kullanarak verileri dönüştürebilir ve toplayabilirsiniz.

Birçok fonksiyon hem veri kod dosyalarında hem de grafik ifadelerinde aynı şekilde kullanılabilir, ancak bunun bazı istisnaları vardır:

- Bazı fonksiyonlar yalnızca veri kod dosyalarında kullanılabilir. Bunlar "kod fonksiyonu" olarak ifade edilir.
- Bazı fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir. Bunlar "grafik fonksiyonu" olarak ifade edilir.
- Bazı fonksiyonlar hem veri kod dosyalarında hem de grafik ifadelerinde kullanılabilir, ancak parametreler ve uygulama bakımından farklar vardır. Bunlar, "kod fonksiyonu" veya "grafik fonksiyonu" olarak ifade edilen ayrı konu başlıklarında açıklanmaktadır.

5.1 Sunucu tarafı uzantılar (SSE) için analiz bağlantıları

Analiz bağlantıları tarafından kullanılabilen işlevler, yalnızca analiz bağlantıları yapılandırıldığı ve Qlik Sense başlatıldığı takdirde görünebilir.

Analiz bağlantılarını QMC üzerinden yapılandırırsınız. Qlik Sense sitelerini yönetme kılavuzundaki "Analiz bağlantısı oluşturma" bölümünü inceleyin.

Qlik Sense Desktop uygulamasında, analiz bağlantılarını yapılandırmak için *Settings.ini* dosyasını düzenlemeniz gerekir. Qlik Sense Desktop kılavuzundaki "Qlik Sense Desktop uygulamasında analiz bağlantılarını yapılandırma" konusunu inceleyin.

5.2 Toplama işlevleri

Toplama işlevleri olarak bilinen işlev ailesi, girdi olarak birden çok alan değeri alan ve grup başına tek bir sonuç döndüren işlevlerden oluşur; burada gruplandırma, kod deyiminde bir grafik boyutu veya bir **group by** cümlesi tarafından tanımlanır.

Toplama işlevleri arasında **Sum()**, **Count()**, **Min()**, **Max()** ve daha birçok işlev yer alır.

Çoğu toplama işlevi hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir, ancak söz dizimi farklılık gösterir.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri kod dosyasında toplama işlevleri kullanma

Toplama işlevleri yalnızca **LOAD** ve **SELECT** deyimleri içinde kullanılabilir.

Grafik ifadelerinde toplama işlevleri kullanma

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Toplama işlevi, seçim ile tanımlanan olası kayıtlar kümesi üzerinden toplanır. Bununla birlikte, set analizinde set ifadesi kullanılarak alternatif bir kayıt kümesi tanımlanabilir.

Toplamaları hesaplama

Toplama, belirli bir tablonun kayıtları üzerinde döngü yaparak, içerdiği kayıtları toplar. Örneğin, **Count** (<Field>), <Field> öğesinin bulunduğu tablodaki kayıtların sayısını sayar. Yalnızca farklı alan değerlerini toplamak istiyorsanız, **Count**(**distinct** <Field>) örneğindeki gibi **distinct** tümcesini kullanmanız gerekir.

Toplama işlevi farklı tablolardan alanlar içeriyorsa, işlevi kurucu alanların tablolarının çapraz ürün kayıtları üzerinde döngü yapar. Bunun performans açısından olumsuz bir yanı vardır ve bu nedenle, özellikle de büyük miktarda veriye sahip olduğunuzda, bu tür toplamalardan kaçınılmalıdır.

Anahtar alanların toplanması

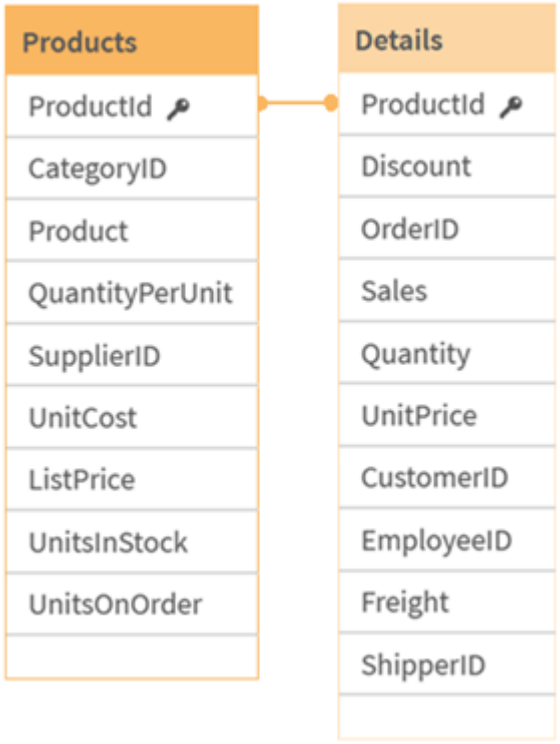
Toplamaların hesaplanma şekli, toplama için hangi tablonun kullanılması gerektiği net olmadığı için anahtar alanları toplayamayacağınız anlamına gelir. Örneğin, <Key> alanı iki tabloyu birbirine bağlıyorsa, **Count**(<Key>) ifadesinin birinci tablonun kayıt sayısını mı yoksa ikinci tablonun kayıt sayısını mı döndüreceği net değildir

Ancak **distinct** cümlesini kullanırsanız toplama iyi tanımlanmıştır ve hesaplanabilir.

Bu nedenle, bir toplama işlevi içinde **distinct** cümlesi olmadan bir anahtar alan kullanırsanız Qlik Sense anlamsız olabilecek bir sayı döndürecektir. Çözüm ya **distinct** cümlesini ya da anahtarın bir kopyasını (yalnızca tek bir tabloda bulunan bir kopya) kullanmaktır.

Örneğin, aşağıdaki tablolarda ProductID, tablolar arasındaki anahtardır.

Ürünler ve Ayrıntılar tabloları arasındaki ProductID anahtarı



Count(ProductID), Products tablosunda (ürün başına yalnızca bir kayıt içerir, ProductID birincil anahtardır) veya Details tablosunda (büyük olasılıkla ürün başına birkaç kayıt içerir) sayılabilir. Farklı ürünlerin sayısını saymak istiyorsanız Count(distinct ProductID) kullanmalısınız. Belirli bir tablodaki satır sayısını saymak istiyorsanız anahtarı kullanmamalısınız.

Temel toplama işlevleri

Temel toplama işlevlerine genel bakış

Temel toplama işlevleri, en yaygın toplama işlevlerinin oluşturduğu gruptur.

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Veri kod dosyasında temel toplama işlevleri

FirstSortedValue

FirstSortedValue(); en düşük birim fiyatına sahip ürünün adı gibi **sort_weight** bağımsız değişkeninin sıralamasının sonucuna karşılık gelen **value** içinde belirtilmiş ifadedeki değeri döndürür. Sıralama düzenindeki n. değer **rank** içinde belirtilebilir. Birden fazla sonuç değeri, belirtilen **rank** için aynı **sort_weight** öğesini paylaşıyorsa fonksiyon NULL döndürür. Sıralanan değerler bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinde yinelenir veya **group by** cümlesi tanımlanmazsa tüm veri kümesi çapında toplanır.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en yüksek sayısal değerini bulur. Bir **rank** n belirtilmesiyle n. en yüksek değer bulunabilir.

```
Max ( expression[, rank])
```

Min

Min(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en düşük sayısal değerini döndürür. Bir **rank** n belirtilmesiyle n. en düşük değer bulunabilir.

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

Mode

Mode(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en yaygın olarak geçen değerini (mod değeri) döndürür. **Mode()** fonksiyonu hem sayısal değerleri hem de metin değerlerini döndürebilir.

```
Mode (expression )
```

Only

Only(), toplanmış verilerde yalnızca bir olası sonuç varsa bir değer döndürür. Kayıt bir değer içeriyorsa bu değer döndürülür, aksi halde NULL döndürülür. Birden fazla kayıt üzerinde değerlendirme yapmak için **group by** cümlesini kullanın. **Only()** fonksiyonu sayısal değerleri ve metin değerlerini döndürebilir.

```
Only (expression )
```

Sum

Sum(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış değerlerin toplamını döndürür.

```
Sum ([distinct]expression)
```

Grafik ifadelerinde temel toplama işlevleri

Grafik toplama işlevleri yalnızca grafik ifadelerindeki alanlarda kullanılabilir. Bir toplama işlevinin bağımsız değişken ifadesi, başka bir toplama işlevini içermemelidir.

FirstSortedValue

FirstSortedValue(); en düşük birim fiyatına sahip ürünün adı gibi **sort_weight** bağımsız değişkeninin sıralamasının sonucuna karşılık gelen **value** içinde belirtilmiş ifadedeki değeri döndürür. Sıralama düzenindeki n. değer **rank** içinde belirtilebilir. Birden fazla sonuç değeri, belirtilen **rank** için aynı **sort_weight** öğesini paylaşıyorsa fonksiyon NULL döndürür.

```
FirstSortedValue - grafik fonksiyonu([{SetExpression}] [DISTINCT] [TOTAL
[<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max(), toplanmış verilerin en yüksek değerini bulur. Bir **rank** n belirtilmesiyle n. en yüksek değer bulunabilir.

```
Max - grafik fonksiyonuMax(), toplanmış verilerin en yüksek değerini bulur.
Bir rank n belirtilmesiyle n. en yüksek değer bulunabilir. Ayrıca, Max
fonksiyonu ile benzer işlevselliğe sahip FirstSortedValue ve rangemax
fonksiyonlarına bakmak isteyebilirsiniz. Max([{SetExpression}] [TOTAL [<fld
```

```
{,fld}>]] expr [,rank]) sayısal Bağımsız DeğişkenlerBağımsız
DeğişkenAçıklamaexprHesaplanacak verileri içeren ifade veya
alan.rankVarsayılan rank değeri 1'dir ve bu en düşük yüksek karşılık gelir.
rank değeri 2 olarak belirtildiğinde en yüksek ikinci değer döndürülür. rank
değeri 3 olursa en yüksek üçüncü değer döndürülür ve bu böyle devam
eder.SetExpressionToplama işlevi, varsayılan olarak, seçim tarafından
tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi
ile alternatif bir kayıt kümesi tanımlanabilir. TOTALTOTAL sözcüğü, fonksiyon
bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal
değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler
üzerinden yapılır; yani grafik boyutlarını göz ardı eder. TOTAL
niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir
veya daha çok alan adının geldiği TOTAL [<fld {.fld}>] niteleyicisini
kullanarak toplam olası değerlerin bir alt kümesini
oluşturursunuz. VerilerCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Örnekler ve sonuçlarÖrneklerSonuçlarMax
(UnitSales)10; çünkü UnitSales içinde en yüksek değer budur.Bir siparişin
değeri, satılan birim sayısı (UnitSales) ile birim fiyatın çarpımından
hesaplanır.Max(UnitSales*UnitPrice)150; çünkü tüm olası (UnitSales)*
(UnitPrice) değerlerinin hesaplanması sonucunda en yüksek değer budur.Max
(UnitSales, 2)9; yani en yüksek ikinci değer.Max(TOTAL UnitSales)10; çünkü
TOTAL niteleyicisi, grafik boyutlarını göz ardı ederek en yüksek olası
değerin bulunması anlamına gelir. Boyut olarak Customer öğesini içeren bir
grafikte, TOTAL niteleyicisi her bir müşteri için maksimum UnitSales yerine,
tüm veri kümesi genelinde maksimum değerin getirilmesini sağlar. Customer B
seçimini yapın.Max({1} TOTAL UnitSales)Set Analysis ifadesi {1} yapılan
seçimden bağımsız olarak ALL şeklinde değerlendirilecek kayıt kümesini
tanımladığından 10 (yapılan seçimden bağımsız olarak).Örneklerde kullanılan
veriler: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(), toplanmış verilerin en düşük değerini bulur. Bir **rank** n belirtilmesiyle n. en düşük değer bulunabilir.

```
Min - grafik fonksiyonu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr [,rank])
```

Mode

Mode(), toplanmış verilerde en yaygın olarak geçen değeri (mod değeri) bulur. **Mode()** fonksiyonu hem metin değerlerini hem de sayısal değerleri işleyebilir.

```
Mode - grafik fonksiyonu ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only(), toplanmış verilerde yalnızca bir olası sonuç varsa bir değer döndürür. Örneğin, birim fiyatın =9 olduğu tek bir ürün arandığında, birden fazla ürünün birim fiyatı 9 ise NULL döndürülür.

```
Only - grafik fonksiyonu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum

Sum(), toplanmış veriler genelinde ifadenin veya alanın verdiği değerlerin toplamını hesaplar.

```
Sum - grafik fonksiyonu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

FirstSortedValue(); en düşük birim fiyatına sahip ürünün adı gibi **sort_weight** bağımsız değişkeninin sıralamasının sonucuna karşılık gelen **value** içinde belirtilmiş ifadedeki değeri döndürür. Sıralama düzenindeki n. değer **rank** içinde belirtilebilir. Birden fazla sonuç değeri, belirtilen **rank** için aynı **sort_weight** öğesini paylaşıyorsa fonksiyon NULL döndürür. Sıralanan değerler bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinde yinelenir veya **group by** cümlesi tanımlanmazsa tüm veri kümesi çapında toplanır.

Söz Dizimi:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value Expression	Fonksiyon, **sort_weight** sıralamasının sonucuna karşılık gelen **value** ifadesi değerini bulur.
sort-weight Expression	Sıralanacak verileri içeren ifade. İlk (en düşük) **sort_weight** değeri bulunur (**value** ifadesinin karşılık gelen değeri bu değerden belirlenir). **sort_weight** öğesinin önüne bir eksi işareti koyarsanız, fonksiyon son (en yüksek) sıralanan değeri döndürür.
rank Expression	1'den büyük bir **rank** "n" belirttiğinizde n. sıralanan değeri alırsınız.
distinct	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örnekleri

Örnek	Sonuç

`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`
	Fonksiyon, UnitSales değerini, en küçük UnitSales bulunan Customer değerini arayarak küçükten büyüğe sıralar.
	
	Çünkü CC değeri Astrida müşterisi için en küçük siparişe (UnitSales değeri = 2) karşılık gelir. AA değeri Betacab müşterisi için en küçük siparişe (4) karşılık gelir; AA değeri Canutility müşterisi için en küçük siparişe (8) karşılık gelir ve DD değeri de Divadip. müşterisi için en küçük siparişe (10) karşılık gelir.

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:
 	
`LOAD Customer,FirstSortedValue(Product, -UnitSales)`
`as MyProductWithLargestOrderByCustomer Resident`
`Temp Group By Customer;`
	`Customer`
	`MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip –`
	sort_weight bağımsız değişkeninin önünde bir eksi işareti bulunduğundan fonksiyon önce en büyük değeri sıralar.
	
	Çünkü AA değeri Astrida müşterisi için en büyük siparişe (UnitSales değeri: 18) karşılık gelir; DD değeri Betacab müşterisi için en büyük siparişe (12) karşılık gelir ve CC değeri de Canutility müşterisi için en büyük siparişe (13) karşılık gelir. Divadip müşterisinin en büyük sipariş (16) için iki aynı değeri vardır; dolayısıyla bu bir null sonuç üretir.

Örnek	Sonuç

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	distinct niteleyicisinin kullanılması dışında önceki örnekle aynıdır. Bu niteleyici Divadip için çoğaltma sonucun göz ardı edilerek null olmayan bir değer döndürülmesini sağlar.

FirstSortedValue - grafik fonksiyonu

FirstSortedValue(); en düşük birim fiyatına sahip ürünün adı gibi **sort_weight** bağımsız değişkeninin sıralamasının sonucuna karşılık gelen **value** içinde belirtilmiş ifadedeki değeri döndürür. Sıralama düzenindeki n. değer **rank** içinde belirtilebilir. Birden fazla sonuç değeri, belirtilen **rank** için aynı **sort_weight** öğesini paylaşıyorsa fonksiyon NULL döndürür.

Söz Dizimi:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Çıkış alanı. Fonksiyon, **sort_weight** sıralamasının sonucuna karşılık gelen **value** ifadesi değerini bulur.
sort_weight	Giriş alanı. Sıralanacak verileri içeren ifade. İlk (en düşük) **sort_weight** değeri bulunur (**value** ifadesinin karşılık gelen değeri bu değerden belirlenir). **sort_weight** öğesinin önüne bir eksi işareti koyarsanız, fonksiyon son (en yüksek) sıralanan değeri döndürür.
rank	1'den büyük bir **rank** "n" belirttiğinizde n. sıralanan değeri alırsınız.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.

Bağımsız Değişken	Açıklama
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Veriler

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

Örnekler ve sonuçlar

Örnek	Sonuç
`firstsortedvalue (Product, UnitPrice)`	BB; yani, `UnitPrice` değeri (9) en düşük `Product`.
`firstsortedvalue (Product, UnitPrice, 2)`	BB; yani, `UnitPrice` değeri (10) en düşük ikinci `Product`.
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab; yani, `UnitPrice` değeri (20) en yüksek ikinci `Product` sahibi `Customer`.
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL; çünkü aynı rank (en düşük üçüncü) `UnitPrice` (15) ile iki `Customer` değeri (Astrida ve Canutility) var. Beklenmedik null sonuçları oluşmamasını sağlamak için distinct niteleyicisini kullanın.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility; yani `UnitPrice` ile `UnitSales` çarpımı (120) olarak en yüksek ikinci satış emri değerine sahip `Customer`.

Örneklerde kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en yüksek sayısal değerini bulur. Bir **rank** n belirtilmesiyle n. en yüksek değer bulunabilir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
rank Expression	Varsayılan **rank** değeri 1'dir ve bu en düşük yüksek karşılık gelir. **rank** değeri 2 olarak belirtildiğinde en yüksek ikinci değer döndürülür. **rank** değeri 3 olursa en yüksek üçüncü değer döndürülür ve bu böyle devam eder.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Örnek:

```
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;
```

Sonuç tablosu

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Örnek:

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Sonuç tablosu

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - grafik fonksiyonu

Max(), toplanmış verilerin en yüksek değerini bulur. Bir **rank** n belirtilmesiyle n. en yüksek değer bulunabilir.

*Ayrıca, **Max** fonksiyonu ile benzer işlevselliğe sahip **FirstSortedValue** ve **rangemax** fonksiyonlarına bakmak isteyebilirsiniz.*

Söz Dizimi:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
rank	Varsayılan **rank** değeri 1'dir ve bu en düşük yüksek karşılık gelir. **rank** değeri 2 olarak belirtildiğinde en yüksek ikinci değer döndürülür. **rank** değeri 3 olursa en yüksek üçüncü değer döndürülür ve bu böyle devam eder.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Veriler

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

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Max(UnitSales)`	10; çünkü `UnitSales` içinde en yüksek değer budur.
Bir siparişin değeri, satılan birim sayısı (`UnitSales`) ile birim fiyatın çarpımından hesaplanır. `Max (UnitSales*UnitPrice)`	150; çünkü tüm olası (`UnitSales`)*(`UnitPrice`) değerlerinin hesaplanması sonucunda en yüksek değer budur.
`Max(UnitSales, 2)`	9; yani en yüksek ikinci değer.
`Max(TOTAL UnitSales)`	10; çünkü TOTAL niteleyicisi, grafik boyutlarını göz ardı ederek en yüksek olası değerin bulunması anlamına gelir. Boyut olarak Customer öğesini içeren bir grafikte, TOTAL niteleyicisi her bir müşteri için maksimum UnitSales yerine, tüm veri kümesi genelinde maksimum değerin getirilmesini sağlar.
`Customer B` seçimini yapın. `Max({1} TOTAL UnitSales)`	Set Analysis ifadesi {1} yapılan seçimden bağımsız olarak ALL şeklinde değerlendirilecek kayıt kümesini tanımladığından 10 (yapılan seçimden bağımsız olarak).

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *FirstSortedValue - grafik fonksiyonu (page 215)*
- *RangeMax (page 716)*

Min

Min(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en düşük sayısal değerini döndürür. Bir **rank** n belirtilmesiyle n. en düşük değer bulunabilir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
rank Expression	**rank** öğesinin varsayılan değeri 1'dir ve bu en düşük değere karşılık gelir. **rank** değeri 2 olarak belirtildiğinde en düşük ikinci değer döndürülür. **rank** değeri 3 olursa en düşük üçüncü değer döndürülür ve bu böyle devam eder.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Örnek:

```
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;
```

Sonuç tablosu

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Örnek:

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Sonuç tablosu

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - grafik fonksiyonu

Min(), toplanmış verilerin en düşük değerini bulur. Bir **rank** n belirtilmesiyle n. en düşük değer bulunabilir.

*Ayrıca, **Min** fonksiyonu ile benzer işlevselliğe sahip **FirstSortedValue** ve **rangemin** fonksiyonlarına bakmak isteyebilirsiniz.*

Söz Dizimi:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
rank	**rank** öğesinin varsayılan değeri 1'dir ve bu en düşük değere karşılık gelir. **rank** değeri 2 olarak belirtildiğinde en düşük ikinci değer döndürülür. **rank** değeri 3 olursa en düşük üçüncü değer döndürülür ve bu böyle devam eder.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Veriler

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

Min() fonksiyonu, ifadenin sağladığı değerler dizisinden NULL olmayan bir değer (varsa) döndürmelidir. Örneklerde, verilerde NULL değerler olduğundan, fonksiyon ifadeden ilk NULL olmayan değeri döndürür.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Min(UnitSales)`	2; çünkü `UnitSales` içinde NULL olmayan en düşük değer budur.
Bir siparişin değeri, satılan birim sayısı (`UnitSales`) ile birim fiyatın çarpımından hesaplanır. `Min (UnitSales*UnitPrice)`	40; çünkü tüm olası (`UnitSales`)*(`UnitPrice`) değerlerinin hesaplanması sonucunda NULL olmayan en düşük değer budur.
`Min(UnitSales, 2)`	4; yani, en düşük ikinci değer (NULL değerlerden sonra).
`Min(TOTAL UnitSales)`	2; çünkü TOTAL niteleyicisi, grafik boyutlarını göz ardı ederek en düşük olası değerin bulunması anlamına gelir. Boyut olarak Customer öğesini içeren bir grafikte, TOTAL niteleyicisi her bir müşteri için minimum UnitSales yerine, tüm veri kümesi genelinde minimum değerin döndürülmesini sağlar.

Örnekler	Sonuçlar

Customer B seçimini yapın.
 	
`Min({1}`
`TOTAL UnitSales)`
	2, (yapılan Customer B seçiminden bağımsız olarak).
	
	Set Analysis ifadesi {1}, yapılan seçimden bağımsız olarak ALL şeklinde değerlendirilecek kayıt kümesini tanımlar.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *FirstSortedValue - grafik fonksiyonu (page 215)*
- *RangeMin (page 719)*

Mode

Mode(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış verilerin en yaygın olarak geçen değerini (mod değeri) döndürür. **Mode()** fonksiyonu hem sayısal değerleri hem de metin değerlerini döndürebilir.

Söz Dizimi:

```
Mode ( expr)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Birden fazla değer eşit şekilde yaygın olarak mevcutsa NULL döndürülür.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örnekleri

Örnek	Sonuç
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 çünkü AA birden fazla satılan tek üründür.

Mode - grafik fonksiyonu

Mode(), toplanmış verilerde en yaygın olarak geçen değeri (mod değeri) bulur. **Mode()** fonksiyonu hem metin değerlerini hem de sayısal değerleri işleyebilir.

Söz Dizimi:

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

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Veriler

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

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Mode(UnitPrice)` `Customer A` seçimini yapın.	15; çünkü `UnitSales` içinde en yaygın olarak görülen değer budur. NULL (-) döndürür. Diğerlerinden daha sık görülen tek bir değer yok.
`Mode(Product)` `Customer A` seçimini yapın	AA; çünkü `Product` içinde en yaygın olarak görülen değer budur. NULL (-) döndürür. Diğerlerinden daha sık görülen tek bir değer yok.
`Mode` `(TOTAL UnitPrice)`	15; çünkü TOTAL niteleyicisi, grafik boyutlarını göz ardı etse bile en yaygın olarak görülen değerin halen 15 olduğu anlamına gelir.
`Customer B` seçimini yapın. `Mode({1}` `TOTAL UnitPrice)`	15, (yapılan seçimden bağımsız olarak); çünkü Set Analysis ifadesi {1} yapılan seçimden bağımsız olarak ALL şeklinde değerlendirilecek kayıt kümesini tanımlar.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *Avg - grafik fonksiyonu (page 270)*
- *Median - grafik fonksiyonu (page 307)*

Only

Only(), toplanmış verilerde yalnızca bir olası sonuç varsa bir değer döndürür. Kayıt bir değer içeriyorsa bu değer döndürülür, aksi halde NULL döndürülür. Birden fazla kayıt üzerinde değerlendirme yapmak için **group by** cümlesini kullanın. **Only()** fonksiyonu sayısal değerleri ve metin değerlerini döndürebilir.

Söz Dizimi:

```
Only ( expr )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

```
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;
```

Sonuç tablosu

Customer	MyUniqIDCheck
Astrida	1 Çünkü sadece Astrida müşterisi, CustomerID öğesini de içeren eksiksiz kayıtlara sahiptir.

Only - grafik fonksiyonu

Only(), toplanmış verilerde yalnızca bir olası sonuç varsa bir değer döndürür. Örneğin, birim fiyatın =9 olduğu tek bir ürün arandığında, birden fazla ürünün birim fiyatı 9 ise NULL döndürülür.

Söz Dizimi:

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

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnek verilerde birden fazla olası değer varsa, NULL sonucu istediğiniz durumlarda Only() fonksiyonunu kullanın.

Örnekler ve sonuçlar:

Veriler

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

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Only({<UnitPrice={9}>} Product)`	BB; çünkü `UnitPrice` değeri '9' olan tek `Product` budur.
`Only({<Product={DD}>} Customer)`	Betacab; çünkü 'DD' adında bir `Product` satan tek `Customer` budur.
`Only({<UnitPrice={20}>} UnitSales)`	`UnitPrice` değeri 20 olan `UnitSales` sayısı 2'dir; çünkü `UnitPrice` =20 olan tek bir `UnitSales` değeri vardır.
`Only({<UnitPrice={15}>} UnitSales)`	NULL; çünkü `UnitPrice` =15 olan iki `UnitSales` değeri vardır.

Örneklerde kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış değerlerin toplamını döndürür.

Söz Dizimi:

```
sum ( [ distinct] expr)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
distinct	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.
expr Expression	Hesaplanacak verileri içeren ifade veya alan.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

```
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;
```

Sonuç tablosu

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - grafik fonksiyonu

Sum(), toplanmış veriler genelinde ifadenin veya alanın verdiği değerlerin toplamını hesaplar.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir. *DISTINCT niteleyicisi desteklenmesine karşın, bazı veriler ihmal edildiğinde okuyucuyu yanlış yönlendirerek toplam değerin gösterildiğini düşünmesine neden olabileceğinden, bu niteleyiciyi kullanırken çok dikkatli olun.*
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Veriler

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15

Customer	Product	UnitSales	UnitPrice
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Sum(UnitSales)`	38. `UnitSales` içindeki değerlerin toplamı.
`Sum(UnitSales*UnitPrice)`	505. Tüm `UnitPrice` ile `UnitSales` çarpımlarının toplamı.
`Sum (TOTAL UnitSales*UnitPrice)`	Hem tablodaki tüm satırlar hem de toplam için 505; çünkü TOTAL niteleyicisi, grafik boyutlarını göz ardı ederek toplamın halen 505 olduğu anlamına gelir.
`Customer B` seçimini yapın. `Sum({1} TOTAL UnitSales*UnitPrice)`	Set Analysis ifadesi {1} yapılan seçimden bağımsız olarak ALL şeklinde değerlendirilecek kayıt kümesini tanımladığından 505 (yapılan seçimden bağımsız olarak).

Örneklerde kullanılan veriler:

```
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 '|');
```

Sayaç toplama işlevleri

Sayaç toplama işlevleri, veri kod dosyasında bir dizi kayıt üzerinde veya grafik boyutunda bir dizi değer üzerinde bir ifadenin çeşitli türlerde sayımlarını döndürür.

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Veri kod dosyasında sayaç toplama işlevleri

Count

Count(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış değerlerin sayısını döndürür.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış eksik değerlerin sayısını döndürür.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış NULL değerlerin sayısını döndürür.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadede bulunan sayısal değerlerin sayısını döndürür.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış sayısal olmayan alan değerlerinin sayısını döndürür.

```
TextCount ([ distinct ] expression)
```

Grafik ifadelerinde sayaç toplama işlevleri

Aşağıdaki sayaç toplama işlevleri grafiklerde kullanılabilir:

Count

Count(), her bir grafik boyutunda değerlerin (metin ve sayısal) sayısını toplamak için kullanılır.

```
Count - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

MissingCount(), her bir grafik boyutunda eksik değerlerin sayısını toplamak için kullanılır. Eksik değerlerin tümü sayısal olmayan değerlerdir.

```
MissingCount - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount(), her bir grafik boyutunda NULL değerlerin sayısını toplamak için kullanılır.

```
NullCount - grafik fonksiyonu({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount
NumericCount(), her bir grafik boyutunda sayısal değerlerin sayısını toplar.

```
NumericCount - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount
TextCount(), her bir grafik boyutunda sayısal olmayan alan değerlerinin sayısını toplamak için kullanılır.

```
TextCount - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış değerlerin sayısını döndürür.

Söz Dizimi:

```
Count( [distinct ] expr)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örnekleri

Örnek	Sonuç
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 Customer boyutu sayfadaki tabloya dahil edildiği sürece. Aksi takdirde OrdersByCustomer için sonuç 3, 2 olur.
Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa: LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
İlk örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa: LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 Aynı değere sahip iki OrderNumber değeri olduğundan 1 ve bir null değer.

Count - grafik fonksiyonu

Count(), her bir grafik boyutunda değerlerin (metin ve sayısal) sayısını toplamak için kullanılır.

Söz Dizimi:

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

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Bağımsız Değişken	Açıklama
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

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

Aşağıdaki örneklerde, belirtilen durumlar dışında tüm müşterilerin seçildiği varsayılmaktadır.

Örnekler ve sonuçlar

Örnek	Sonuç
`Count(OrderNumber)`	10; çünkü OrderNumber için değer bulundurabilecek 10 alan vardır ve tüm kayıtlar (boş olanlar bile) sayılır. *"0" boş bir hücre olarak değil, bir değer olarak kabul edilir. Ancak, bir boyut için hesaplamanın toplamı 0 ise bu boyut grafiklere dahil edilmeyecektir.*
`Count(Customer)`	10; çünkü Count fonksiyonu tüm alanlardaki oluşların sayısını değerlendirir.
`Count(DISTINCT [Customer])`	4; çünkü Distinct niteleyicisi kullanıldığında Count yalnızca benzersiz oluşları değerlendirir.
Canutility adlı müşterinin seçildiği varsayıldığında `Count(OrderNumber)/Count ({1} TOTAL OrderNumber)`	0,2; çünkü bu ifade, seçili müşteriden gelen sipariş sayısını tüm müşterilerden gelen siparişlerin yüzdesi olarak döndürür. Bu durumda sonuç 2 / 10 olur.
Astrida ve Canutility adlı müşterilerin seçildiği varsayıldığında `Count(TOTAL <Product> OrderNumber)`	5; çünkü yalnızca seçili müşteriler için verilen ürün siparişlerinin sayısı budur ve boş hücreler sayılmaktadır.

Örneklerde kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış eksik değerlerin sayısını döndürür.

Söz Dizimi:

```
MissingCount ( [ distinct ] expr)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örnekleri

Örnek	Sonuç

`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`
	
	İkinci deyim şunu verir:
	
	`TotalMissingCount`
	`3`
	(bu boyutu içeren bir tabloda).

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:
 	
`LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;`
	`TotalMissingCountDistinct`
	`1`
	Çünkü değeri eksik tek bir OrderNumber vardır.

MissingCount - grafik fonksiyonu

MissingCount(), her bir grafik boyutunda eksik değerlerin sayısını toplamak için kullanılır. Eksik değerlerin tümü sayısal olmayan değerlerdir.

Söz Dizimi:

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

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

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

Customer	Product	OrderNumber	UnitSales	Unit Price
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Örnekler ve sonuçlar

Örnek	Sonuç
`MissingCount([OrderNumber])`	3; çünkü 10 OrderNumber alandan 3 tanesi boştur. *"0" boş bir hücre olarak değil, bir değer olarak kabul edilir. Ancak, bir boyut için hesaplamanın toplamı 0 ise bu boyut grafiklere dahil edilmeyecektir.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	İfade, seçili müşteriden gelen eksik sipariş sayısını tüm müşterilerden gelen eksik siparişlerin kesri olarak döndürür. Tüm müşteriler için toplam 3 eksik OrderNumber değeri vardır. O halde, Product değeri eksik olan her bir Customer için sonuç 1/3 olur.

Örnekte kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış NULL değerlerin sayısını döndürür.

Söz Dizimi:

```
NullCount ( [ distinct ] expr)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örnekleri

Örnek	Sonuç

`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`
	
	İkinci deyim şunu verir:
	`TotalNullCount`
	`1`
	(bu boyutu içeren bir tabloda), çünkü null değer içeren tek bir kayıt vardır.

NullCount - grafik fonksiyonu

NullCount(), her bir grafik boyutunda NULL değerlerin sayısını toplamak için kullanılır.

Söz Dizimi:

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

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
set_ expression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Sonuç
NullCount ([OrderNumber])	1; çünkü satır içi **LOAD** deyiminde NullInterpret kullanarak bir null değer tanıttık.

Örnekte kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadede bulunan sayısal değerlerin sayısını döndürür.

Söz Dizimi:

```
NumericCount ( [ distinct ] expr)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Kod örneği

Örnek	Sonuç
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	İkinci deyim şunu verir: TotalNumericCount 7 (bu boyutu içeren bir tabloda).
Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Bir diğerini çoğaltan bir OrderNumber bulunduğundan, sonuç çoğaltılmayan 6 olur.

Örnek:

```
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;
```

Sonuç tablosu

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - grafik fonksiyonu

NumericCount(), her bir grafik boyutunda sayısal değerlerin sayısını toplar.

Söz Dizimi:

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

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
set_ expression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

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

Aşağıdaki örneklerde, belirtilen durumlar dışında tüm müşterilerin seçildiği varsayılmaktadır.

Örnekler ve sonuçlar

Örnek	Sonuç
`NumericCount ([OrderNumber])`	7; çünkü OrderNumber içinde 10 alanın üçü boştur. *"0" boş bir hücre olarak değil, bir değer olarak kabul edilir. Ancak, bir boyut için hesaplamanın toplamı 0 ise bu boyut grafiklere dahil edilmeyecektir.*
`NumericCount ([Product])`	0; çünkü tüm ürün adları metinde yer almaktadır. Normalde bunu, hiçbir metin alanına sayısal içerik verilmediğini kontrol etmek için kullanabilirsiniz.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Tekil sayısal sıra numaralarının tümünü sayar ve bu değeri sayısal ve sayısal olmayan sıra numaralarının sayısına böler. Tüm alan değerlerinin sayısal olması durumunda bu değer 1 olacaktır. Normalde bunu, tüm alan değerlerinin sayısal olduğunu kontrol etmek için kullanabilirsiniz. Örnekte 8 tekil sayısal ve sayısal olmayan değer içinde OrderNumber için 7 tekil sayısal değer vardır; bu nedenle ifade 0,875 döndürür.

Örnekte kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, ifadedeki toplanmış sayısal olmayan alan değerlerinin sayısını döndürür.

Söz Dizimi:

```
TextCount ( [ distinct ] expr)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr Expression	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Örnek:

```
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;
```

Sonuç tablosu

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Örnek:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Sonuç tablosu

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - grafik fonksiyonu

TextCount(), her bir grafik boyutunda sayısal olmayan alan değerlerinin sayısını toplamak için kullanılır.

Söz Dizimi:

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

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

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

Örnekler ve sonuçlar

Örnek	Sonuç
`TextCount ([Product])`	10; çünkü Product sütunundaki 10 alanın tümü metindir. *"0" boş bir hücre olarak değil, bir değer olarak kabul edilir. Ancak, bir boyut için hesaplamanın toplamı 0 ise bu boyut grafiklere dahil edilmeyecektir. Boş hücrelerin metin olmadığı varsayılır ve bunlar TextCount tarafından sayılmaz.*
`TextCount ([OrderNumber])`	3; çünkü boş hücreler sayılır. Normalde bunu, sayısal alanların hiçbirine metin değerleri verilmediğini veya bu alanların sıfır olmadığını kontrol etmek için kullanırsınız.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Tüm tekil Product metin değerlerini (4) sayar ve Product içindeki toplam değer sayısına (10) böler. Sonuç 0,4'tür.

Örnekte kullanılan veriler:

```
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 '|');
```

Finansal toplama işlevleri

Bu bölümde, ödemeler ve nakit akışı ile ilgili finansal işlemlere yönelik toplama işlevleri açıklanmaktadır.

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Veri kod dosyasında finansal toplama işlevleri

IRR

IRR(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen ifadedeki sayılar tarafından temsil edilen nakit akışlarından oluşan bir seri için toplam iç geri dönüş oranını döndürür.

```
IRR (expression)
```

XIRR

XIRR(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **pmt** ve **date** içindeki eşlenmiş sayılar tarafından temsil edilen (dönemsel olması gerekmeyen) nakit akışlarının planı için toplam iç geri dönüş oranını döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV(); bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **value** içindeki sayılar tarafından temsil edilen dönem başına **discount_rate** ve bir gelecekteki ödemeler (negatif değerler) ile gelirler (pozitif değerler) dizisine dayanan bir yatırımın toplam net mevcut değerini döndürür. Ödemelerin ve gelirlerin her bir dönemin sonunda meydana geldiği varsayılır.

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

XNPV

XNPV(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **pmt** ve **date** içindeki eşlenmiş sayılar tarafından temsil edilen (dönemsel olması gerekmeyen) nakit akışlarının planı için toplam net bugünkü değerini döndürür. Rate, her dönem için faiz oranıdır. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

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

Grafik ifadelerinde finansal toplama işlevleri

Bu finansal toplama işlevleri grafiklerde kullanılabilir.

IRR

IRR(), grafik boyutları üzerinde yinelenen **value** ile verilen ifadedeki sayıların temsil ettiği bir nakit akışı serisi için toplam iç geri dönüş oranını döndürür.

```
IRR - grafik fonksiyonu[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV(), grafik boyutları üzerinde yinelenen, **value** içindeki sayıların temsil ettiği bir dizi gelecek ödeme (negatif değerler) ve gelirlere ve dönem başına **discount_rate** değerine dayalı olarak bir yatırımın toplam net bugünkü değerini döndürür. Ödemelerin ve gelirlerin her bir dönemin sonunda meydana geldiği varsayılır.

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

XIRR

XIRR(), grafik boyutları üzerinde yinelenen **pmt** ve **date** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir nakit akışları planı için (mutlaka dönemsel olması gerekmez) toplam iç geri dönüş oranını döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

```
XIRR - grafik fonksiyonu (page 258)([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV(), grafik boyutları üzerinde yinelenen **pmt** ve **date** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir nakit akışları planı için (mutlaka dönemsel olması gerekmez) toplam net bugünkü değeri döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

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

IRR

IRR(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen ifadedeki sayılar tarafından temsil edilen nakit akışlarından oluşan bir seri için toplam iç geri dönüş oranını döndürür.

Yıllık gelirle ilgili olduklarından, bu nakit akışlarının birbirine eşit olmaları gerekmez. Bununla birlikte, nakit akışlarının aylık veya yıllık gibi düzenli aralıklarla meydana gelmesi gerekir. Dahili geri dönüş oranı, düzenli aralıklarda meydana gelen ödemelerden (negatif değerler) ve gelirden (pozitif değerler) oluşan bir yatırım için alınan faiz oranıdır. Hesaplama için fonksiyon en az bir pozitif ve bir negatif değere ihtiyaç duyar.

Söz Dizimi:

```
IRR(value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Yıl	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 - grafik fonksiyonu

IRR(), grafik boyutları üzerinde yinelenen **value** ile verilen ifadedeki sayıların temsil ettiği bir nakit akışı serisi için toplam iç geri dönüş oranını döndürür.

Yıllık gelirle ilgili olduklarından, bu nakit akışlarının birbirine eşit olmaları gerekmez. Bununla birlikte, nakit akışlarının aylık veya yıllık gibi düzenli aralıklarla meydana gelmesi gerekir. Dahili geri dönüş oranı, düzenli aralıklarda meydana gelen ödemelerden (negatif değerlerden) ve gelirden (pozitif değerlerden) oluşan ve bir yatırım için alınan faiz oranıdır. Hesaplamak için, bu fonksiyon en az bir pozitif ve bir negatif değere ihtiyaç duyar.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Hesaplanacak verileri içeren ifade veya alan.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Sonuç
`IRR (Payments)`	0.1634 Ödemelerin tabiatı gereği dönemsel (örneğin, aylık) olduğu varsayılır. *Tarih alanı, ödemelerin yapıldığı tarihleri sağladığınız sürece ödemelerin dönemsel olmayabileceği XIRR örneğinde kullanılır.*

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *XIRR - grafik fonksiyonu (page 258)*
- *Aggr - grafik fonksiyonu (page 410)*

NPV

NPV(); bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **value** içindeki sayılar tarafından temsil edilen dönem başına **discount_rate** ve bir gelecekteki ödemeler (negatif değerler) ile gelirler (pozitif değerler) dizisine dayanan bir yatırımın toplam net mevcut değerini döndürür. Ödemelerin ve gelirlerin her bir dönemin sonunda meydana geldiği varsayılır.

Söz Dizimi:

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

Dönüş verileri türü: sayısal. Sonuç, para için varsayılan sayı biçimine sahiptir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	**discount_rate**, dönem boyunca iskonto oranıdır.
value	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnekler ve sonuçlar

Örnek	Yıl	NPV1_2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,NPV(0.2, Payments) as NPV1_2013 Resident Cashflow Group By Year;	2013	-$540.12

Örnekler ve sonuçlar

Örnek	Yıl	İndirim	NPV2_2013
Önceki örnekte olduğu gibi **Cashflow** tablosunun yüklendiği varsayılırsa: `LOAD Year,NPV(Discount, Payments) as NPV2_2013 Resident Cashflow Group By Year, Discount;` Group By cümlesinin sonuçları Year ve Discount öğelerine göre sıraladığını unutmayın. İlk bağımsız değişken olan discount_rate, belirli bir sayı nedeniyle bir alan (Discount) olarak verilir ve bu nedenle ikinci bir sıralama ölçütü gereklidir. Bir alan farklı değerler içerebilir, bu nedenle toplanmış kayıtlar farklı Year ve Discount değerlerine izin verecek şekilde sıralanmalıdır. ;	2013 2013	0.1 0.2	-$3456.05 $5666.67

NPV - grafik fonksiyonu

NPV(), grafik boyutları üzerinde yinelenen, **value** içindeki sayıların temsil ettiği bir dizi gelecek ödeme (negatif değerler) ve gelirlere ve dönem başına **discount_rate** değerine dayalı olarak bir yatırımın toplam net bugünkü değerini döndürür. Ödemelerin ve gelirlerin her bir dönemin sonunda meydana geldiği varsayılır.

Söz Dizimi:

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

Dönüş verileri türü: sayısal Sonuç, para için varsayılan sayı biçimine sahiptir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	**discount_rate**, dönem boyunca iskonto oranıdır.
value	Hesaplanacak verileri içeren ifade veya alan.

Bağımsız Değişken	Açıklama
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz. **TOTAL** niteleyicisinin ardından açılı ayraçlar içindeki bir veya daha fazla alan adından oluşan bir liste gelebilir. Bu alan adları grafik boyut değişkenlerinin bir alt kümesi olmalıdır. Bu durumda, hesaplama listelenenler dışındaki tüm grafik boyut değişkenlerini göz ardı ederek yapılır; yani listelenen boyut alanlarındaki alan değerlerinin her bir kombinasyonu için bir değer döndürülür. Ayrıca, geçerli anda grafikte bir boyut olmayan alanlar da listeye dahil edilebilir. Bu, boyut alanlarının sabit olmadığı grup boyutları durumunda kullanışlı olabilir. Gruptaki tüm değişkenlerin listelenmesi, detaya inme düzey değişikliği olduğunda fonksiyonun çalışmasına neden olur.

Sınırlamalar:

Bu iç toplamlar **TOTAL** niteleyicisini içermedikçe, **discount_rate** ve **value** öğeleri toplama işlevleri içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Sonuç
`NPV(Discount, Payments)`	-$540.12

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

XNPV - grafik fonksiyonu (page 260)
Aggr - grafik fonksiyonu (page 410)

XIRR

XIRR(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **pmt** ve **date** içindeki eşlenmiş sayılar tarafından temsil edilen (dönemsel olması gerekmeyen) nakit akışlarının planı için toplam iç geri dönüş oranını döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Söz Dizimi:

```
XIRR(pmt, date )
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
pmt	Ödemeler. **date** içinde verilen ödeme planına karşılık gelen nakit akışlarını içeren ifade veya alan.
date	**pmt** içinde verilen nakit akışı ödemelerine karşılık gelen tarih planını içeren ifade veya alan.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnekler ve sonuçlar

Örnek	Yıl	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 - grafik fonksiyonu

XIRR(), grafik boyutları üzerinde yinelenen **pmt** ve **date** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir nakit akışları planı için (mutlaka dönemsel olması gerekmez) toplam iç geri dönüş oranını döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
pmt	Ödemeler. **date** içinde verilen ödeme planına karşılık gelen nakit akışlarını içeren ifade veya alan.
date	**pmt** içinde verilen nakit akışı ödemelerine karşılık gelen tarih planını içeren ifade veya alan.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamlar **TOTAL** niteleyicisini içermedikçe, **pmt** ve **date** öğeleri toplama işlevleri içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Sonuç
`XIRR(Payments, Date)`	0.5385

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *IRR - grafik fonksiyonu (page 252)*
- *Aggr - grafik fonksiyonu (page 410)*

XNPV

XNPV(), bir group by cümlesi tarafından tanımlandığı şekliyle, birden fazla kayıt üzerinde yinelenen **pmt** ve **date** içindeki eşlenmiş sayılar tarafından temsil edilen (dönemsel olması gerekmeyen) nakit akışlarının planı için toplam net bugünkü değerini döndürür. Rate, her dönem için faiz oranıdır. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Söz Dizimi:

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

Dönüş verileri türü: sayısal. Sonuç, para için varsayılan sayı biçimine sahiptir. .

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	**discount_rate**, dönem boyunca iskonto oranıdır.
pmt	Hesaplanacak verileri içeren ifade veya alan.
date	**pmt** içinde verilen nakit akışı ödemelerine karşılık gelen tarih planını içeren ifade veya alan.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnekler ve sonuçlar

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

Örnekler ve sonuçlar

Örnek	Yıl	İndirim	XNPV2_2013
Önceki örnekte olduğu gibi **Cashflow** tablosunun yüklendiği varsayılırsa: LOAD Year,XNPV(Discount, Payments, Date) as XNPV2_2013 Resident Cashflow Group By Year, Discount; Group By cümlesinin sonuçları Year ve Discount öğelerine göre sıraladığını unutmayın. İlk bağımsız değişken olan discount_rate, belirli bir sayı nedeniyle bir alan (Discount) olarak verilir ve bu nedenle ikinci bir sıralama ölçütü gereklidir. Bir alan farklı değerler içerebilir, bu nedenle toplanmış kayıtlar farklı Year ve Discount değerlerine izin verecek şekilde sıralanmalıdır.	2013 2013	0.1 0.2	-$3164.35 $6800.00

XNPV - grafik fonksiyonu

XNPV(), grafik boyutları üzerinde yinelenen **pmt** ve **date** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir nakit akışları planı için (mutlaka dönemsel olması gerekmez) toplam net bugünkü değeri döndürür. Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Söz Dizimi:

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

Dönüş verileri türü: sayısal Sonuç, para için varsayılan sayı biçimine sahiptir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	**discount_rate**, dönem boyunca iskonto oranıdır.
pmt	Ödemeler. **date** içinde verilen ödeme planına karşılık gelen nakit akışlarını içeren ifade veya alan.
date	**pmt** içinde verilen nakit akışı ödemelerine karşılık gelen tarih planını içeren ifade veya alan.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamlar **TOTAL** veya **ALL** niteleyicilerini içermedikçe **discount_rate**, **pmt** ve **date** öğelerinin toplama işlevleri içermemesi gerekir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnek	Sonuç
`XNPV(Discount, Payments, Date)`	-$3164.35

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *NPV - grafik fonksiyonu (page 255)*
- *Aggr - grafik fonksiyonu (page 410)*

İstatistiksel toplama işlevleri

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Veri kod dosyasında istatistiksel toplama işlevleri

Aşağıdaki istatistiksel toplama işlevleri kodlarda kullanılabilir.

Avg

Avg(), **group by** cümlesi tarafından tanımlanan birkaç kayıt içerisinden ifadedeki birleştirilmiş verilerin ortalama değerini bulur.

```
Avg  ([distinct] expression)
```

Correl

Correl(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için toplam korelasyon katsayısını döndürür.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde ifadedeki toplanmış verilerin kapsayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde ifadedeki toplanmış verilerin dışlayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadedeki verilerin basıklığını döndürür.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplam b değerini (y kesişmesini) döndürür.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış serbestlik derecesini döndürür.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Kod fonksiyonu, bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış F istatistiğini ($r^2/(1-r^2)$) döndürür.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplam m değerini (eğim) döndürür.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış r^2 değerini (determinasyon katsayısı) döndürür.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış b değeri standart hatasını döndürür.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış m değeri standart hatasını döndürür.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış y tahmini standart hatasını döndürür.

```
LINEST_SEY (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış regresyon kareleri toplamını döndürür.

```
LINEST_SSREG (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış kalan kareler toplamını döndürür.

```
LINEST_SSRESID (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadedeki değerlerin toplanmış medyanını döndürür.

```
Median (expression)
```

Skew

Skew(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadenin eğriliğini döndürür.

```
Skew ([ distinct] expression)
```

Stdev

Stdev(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifade ile verilen değerlerin standart sapmasını döndürür.

```
Stdev ([distinct] expression)
```

Sterr

Sterr(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen ifadenin temsil ettiği bir değerler dizisi için toplanmış standart hatayı (stdev/sqrt(n)) döndürür.

```
Sterr ([distinct] expression)
```

STEYX

STEYX(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için regresyondaki her bir x değeri için tahmini y değerinin toplam standart hatasını döndürür.

```
STEYX (y-expression, x-expression)
```

Grafik ifadelerinde istatistiksel toplama işlevleri

Aşağıdaki istatistiksel toplama işlevleri grafiklerde kullanılabilir.

Avg

Avg(), grafik boyutları üzerinde yinelenen ifade veya alanın toplanmış ortalamasını döndürür.

```
Avg - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Correl(), iki veri kümesi için toplanmış korelasyon katsayısını döndürür. Korelasyon fonksiyonu veri kümeleri arasındaki ilişkinin bir hesaplamasıdır ve grafik boyutları üzerinde yinelenen (x,y) değer çiftleri için toplanır.

```
Correl - grafik fonksiyonu({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Fractile() grafik boyutları üzerinde yinelenen ifade ile verilen aralıkta toplanmış verilerin kapsayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

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

FractileExc

FractileExc() grafik boyutları üzerinde yinelenen ifade ile verilen aralıkta toplanmış verilerin dışlayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

```
FractileExc - grafik fonksiyonu({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Kurtosis(), grafik boyutları üzerinde yinelenen ifade veya alanda toplanmış veriler aralığının basıklığını bulur.

```
Kurtosis - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

LINEST_B(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ifadeleri ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış b değerini (y kesimi) döndürür.

```
LINEST_R2 - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış serbestlik derecelerini döndürür.

```
LINEST_DF - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış F istatistiğini (r2/(1-r2)) döndürür.

```
LINEST_F - grafik fonksiyonu({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış m değerini (eğim) döndürür.

```
LINEST_M - grafik fonksiyonu({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış r2 değerini (determinasyon katsayısı) döndürür.

```
LINEST_R2 - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait b değeri toplanmış standart hatasını döndürür.

```
LINEST_SEB - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait m değeri toplanmış standart hatasını döndürür.

```
LINEST_SEM - grafik fonksiyonu([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait y tahmini toplanmış standart hatasını döndürür.

```
LINEST_SEY - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış regresyon kareler toplamını döndürür.

```
LINEST_SSREG - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID(); grafik boyutları üzerinde yinelenen **x_value** ve **y_value** tarafından verilen ifadelerdeki eşleştirilmiş sayılarla temsil edilen bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış kalan kareler toplamını döndürür.

```
LINEST_SSRESID - grafik fonksiyonuLINEST_SSRESID(); grafik boyutları üzerinde
yinelenen x_value ve y_value tarafından verilen ifadelerdeki eşleştirilmiş
sayılarla temsil edilen bir koordinat dizisi için y=mx+b denklemiyle
tanımlanan bir doğrusal regresyona ait toplanmış kalan kareler toplamını
döndürür.  LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]]
y_value, x_value[, y0_const[, x0_const]]) sayısal Bağımsız
DeğişkenlerBağımsız DeğişkenAçıklamay_valueHesaplanacak y değerleri aralığını
içeren ifade veya alan.x_valueHesaplanacak x değerleri aralığını içeren ifade
veya alan.y0, x0Regresyon çizgisini belirli bir noktada y ekseninden geçmeye
zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de
x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye
zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en
az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri
çifti yeterli olur. SetExpressionToplama işlevi, varsayılan olarak, seçim
tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set
analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCTFonksiyon bağımsız değişkenlerinden önce DISTINCT sözcüğü varsa
fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan
çoğaltmalar göz ardı edilir. TOTALTOTAL sözcüğü, fonksiyon bağımsız
değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait
olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden
yapılır; yani grafik boyutlarını göz ardı eder. TOTAL niteleyicisinden sonra
grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan
adının geldiği TOTAL [<fld {.fld}>] niteleyicisini kullanarak toplam olası
değerlerin bir alt kümesini oluşturursunuz.Regresyon çizgisini belirli bir
noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri
belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir
```

sabit koordinattan geçmeye zorlanabilir. Bu iç toplamalar TOTAL niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş Aggr işlevini kullanın. Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur. An example of how to use linest functionsavg({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

Median

Median(), grafik boyutları üzerinde yinelenen ifadede toplanmış değerler aralığının medyan değerini döndürür.

Median - grafik fonksiyonu({[SetExpression] [**TOTAL** [<fld{, fld}>]]} expr**)**

MutualInfo

MutualInfo, iki alan arasındaki veya **Aggr()** içindeki toplu değerler arasındaki karşılıklı bilgileri (MI) hesaplar.

MutualInfo - grafik fonksiyonu (page 309){[SetExpression] [**DISTINCT**] [**TOTAL** target, driver [, datatype [, breakdownbyvalue [, samplesize]]]**)**

Skew

Skew(), grafik boyutları üzerinde yinelenen ifadenin veya alanın toplanmış eğriliğini döndürür.

Skew - grafik fonksiyonu{[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{ ,fld}>]]} expr**)**

Stdev

Stdev(), grafik boyutları üzerinde yinelenen ifade veya alanda toplanmış veriler aralığının standart sapmasını bulur.

Stdev - grafik fonksiyonu({[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{, fld}>]]} expr**)**

Sterr

Sterr(), grafik boyutları üzerinde yinelenen ifadede toplanmış değer dizisi için ortalamanın standart hatası değerini (stdev/sqrt(n)) bulur.

Sterr - grafik fonksiyonu({[SetExpression] [**DISTINCT**] [**TOTAL**[<fld{, fld}>]]} expr**)**

STEYX

STEYX(), **y_value** ve **x_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi ile verilen doğrusal regresyonda her bir x değeri için y değerlerini tahmin ederken toplanmış standart hatayı döndürür.

STEYX - grafik fonksiyonu{[SetExpression] [**TOTAL** [<fld{, fld}>]]} y_value, x_value**)**

Avg

Avg(), **group by** cümlesi tarafından tanımlanan birkaç kayıt içerisinden ifadedeki birleştirilmiş verilerin ortalama değerini bulur.

Söz Dizimi:

```
Avg([DISTINCT] expr)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
DISTINCT	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç

`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`
	Hesaplamayı içeren bir tablo oluşturmak suretiyle sayfada bunun kontrolü yapılabilir:
	`Sum(Sales)/12`

Önceki örnekte olduğu gibi **Temp** tablosunun yüklendiği varsayılırsa:
`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`
	Yalnızca tekil değerler sayılır. Toplamı, çoğaltılmayan değerlerin sayısına bölün.

Avg - grafik fonksiyonu

Avg(), grafik boyutları üzerinde yinelenen ifade veya alanın toplanmış ortalamasını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

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

İşlev örnekleri

Örnek	Sonuç
`Avg(Sales)`	Boyutu ve `Customer` `Avg([Sales])` hesaplamasını içeren bir tablo için **Toplamlar** gösteriliyorsa sonuç 2566 olur.
`Avg([TOTAL (Sales))`	Tüm `Customer` değerleri için 53,458333 olur; çünkü TOTAL niteleyicisi boyutların göz ardı edilmesi anlamını taşır.
`Avg (DISTINCT (Sales))`	Toplam için 51,862069 olur; çünkü Distinct niteleyicisinin kullanılması, her bir `Sales` için yalnızca benzersiz `Customer` değerlerinin değerlendirilmesi anlamını taşır.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

Aggr - grafik fonksiyonu (page 410)

Correl

Correl(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için toplam korelasyon katsayısını döndürür.

Söz Dizimi:

```
Correl(value1, value2)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value1, value2	Korelasyon katsayısının hesaplanacağı iki örnek kümeyi içeren ifadeler veya alanlar.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç

`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;`
	`Correl_Salary` boyutunu içeren bir tabloda, veri kod dosyasındaki Correl() hesaplamasının sonucu gösterilecektir: 0,9270611

Correl - grafik fonksiyonu

Correl(), iki veri kümesi için toplanmış korelasyon katsayısını döndürür. Korelasyon fonksiyonu veri kümeleri arasındaki ilişkinin bir hesaplamasıdır ve grafik boyutları üzerinde yinelenen (x,y) değer çiftleri için toplanır.

Söz Dizimi:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value1, value2	Korelasyon katsayısının hesaplanacağı iki örnek kümeyi içeren ifadeler veya alanlar.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnek	Sonuç
`Correl(Age, Salary)`	Boyutu ve `Employee name` `Correl(Age, Salary)` hesaplamasını içeren bir tablo için sonuç 0,9270611 olur. Sonuç yalnızca toplamlar hücresi için görüntülenir.

Örnek	Sonuç
`Correl (TOTAL Age, Salary))`	0.927. Bu ve aşağıdaki sonuçlar, okuma kolaylığı açısından üç ondalık basamağa kadar gösterilir. Gender boyutuyla bir filtre bölmesi oluşturursanız ve bundan seçimler yaparsanız, Female seçildiğinde 0,951 ve Male seçildiğinde 0,939 sonucunu görürsünüz. Bunun nedeni, seçimin diğer Gender değerine ait olmayan tüm sonuçları hariç tutmasıdır.
`Correl({1} TOTAL Age, Salary))`	0.927. Seçimlerden bağımsızdır. Bunun nedeni, {1} set ifadesinin tüm seçimleri ve boyutları göz ardı etmesidir.
`Correl (TOTAL <Gender> Age, Salary))`	Toplam hücrede 0,927, tüm Male değerleri için 0,939 ve tüm Female değerleri için 0,951. Bu, Gender öğesine göre filtre bölmesinde seçim yapmaktan kaynaklanan sonuçlara karşılık gelir.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *Aggr - grafik fonksiyonu (page 410)*
- *Avg - grafik fonksiyonu (page 270)*
- *RangeCorrel (page 707)*

Fractile

Fractile(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde ifadedeki toplanmış verilerin kapsayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

Dışlayıcı fraktili hesaplamak için FractileExc (page 278) kullanabilirsiniz.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Fonksiyon, `rank = fraction * (N-1) + 1` tarafından tanımlandığı şekilde sıralamaya karşılık gelen değeri döndürür; burada `N`, `expr` içindeki değer sayısıdır. `rank`, tamsayı olmayan bir sayı ise en yakın iki değer arasında enterpolasyon yapılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Fraktil hesaplanırken kullanılacak verileri içeren alan veya ifade.
fraction	Hesaplanacak fraktile (kesir olarak ifade edilen yüzdelik dilim) karşılık gelen, 0 ile 1 arasında bir sayı.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç
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;	`Type` ve `MyFractile` boyutlarını içeren bir tabloda, veri kod dosyasındaki Fractile() hesaplamaların sonuçları şöyledir: Type MyFractile Comparison 27.5 Observation 36

Fractile - grafik fonksiyonu

Fractile() grafik boyutları üzerinde yinelenen ifade ile verilen aralıkta toplanmış verilerin kapsayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

Dışlayıcı fraktili hesaplamak için FractileExc - grafik fonksiyonu (page 280) kullanabilirsiniz.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Fonksiyon, `rank = fraction * (N-1) + 1` tarafından tanımlandığı şekilde sıralamaya karşılık gelen değeri döndürür; burada `N`, `expr` içindeki değer sayısıdır. `rank`, tamsayı olmayan bir sayı ise en yakın iki değer arasında enterpolasyon yapılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Fraktil hesaplanırken kullanılacak verileri içeren alan veya ifade.
fraction	Hesaplanacak fraktile (kesir olarak ifade edilen yüzdelik dilim) karşılık gelen, 0 ile 1 arasında bir sayı.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

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

İşlev örnekleri

Örnek	Sonuç
`Fractile (Sales, 0.75)`	Boyutu ve `Customer` `Fractile([Sales])` hesaplamasını içeren bir tablo için **Toplamlar** gösteriliyorsa sonuç 71,75 olur. Bu, `Sales` değerlerinin dağılımında değerlerin %75'inin altına düştüğü noktadır.
`Fractile (TOTAL Sales, 0.75))`	Tüm `Customer` değerleri için 71,75 olur; çünkü TOTAL niteleyicisi boyutların göz ardı edilmesi anlamını taşır.
`Fractile (DISTINCT Sales, 0.75)`	Toplam için 70 olur; çünkü DISTINCT niteleyicisinin kullanılması, her bir `Sales` için yalnızca benzersiz `Customer` değerlerinin değerlendirilmesi anlamını taşır.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

Aggr - grafik fonksiyonu (page 410)

FractileExc

FractileExc(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde ifadedeki toplanmış verilerin dışlayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

Kapsayıcı fraktili hesaplamak için Fractile (page 274) kullanabilirsiniz.

Söz Dizimi:

```
FractileExc(expr, fraction)
```

Dönüş verileri türü: sayısal

Fonksiyon, `rank = fraction * (N-1)` tarafından tanımlandığı şekilde sıralamaya karşılık gelen değeri döndürür; burada `N`, `expr` içindeki değer sayısıdır. `rank`, tamsayı olmayan bir sayı ise en yakın iki değer arasında enterpolasyon yapılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Fraktil hesaplanırken kullanılacak verileri içeren alan veya ifade.
fraction	Hesaplanacak fraktile (kesir olarak ifade edilen yüzdelik dilim) karşılık gelen, 0 ile 1 arasında bir sayı.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç
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;	`Type` ve `MyFractile` boyutlarını içeren bir tabloda, veri kod dosyasındaki FractileExc() hesaplamaların sonuçları şöyledir: Type MyFractile Comparison 28.5 Observation 38

FractileExc - grafik fonksiyonu

FractileExc() grafik boyutları üzerinde yinelenen ifade ile verilen aralıkta toplanmış verilerin dışlayıcı fraktiline (yüzdelik dilim) karşılık gelen değeri bulur.

Kapsayıcı fraktili hesaplamak için Fractile - grafik fonksiyonu (page 276) kullanabilirsiniz.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Fonksiyon, `rank = fraction * (N-1)` tarafından tanımlandığı şekilde sıralamaya karşılık gelen değeri döndürür; burada `N`, `expr` içindeki değer sayısıdır. `rank`, tamsayı olmayan bir sayı ise en yakın iki değer arasında enterpolasyon yapılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Fraktil hesaplanırken kullanılacak verileri içeren alan veya ifade.
fraction	Hesaplanacak fraktile (kesir olarak ifade edilen yüzdelik dilim) karşılık gelen, 0 ile 1 arasında bir sayı.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

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

İşlev örnekleri

Örnek	Sonuç
`FractileExc (Sales, 0.75)`	Boyutu ve `Customer` `FractileExc([Sales])` hesaplamasını içeren bir tablo için **Toplamlar** gösteriliyorsa sonuç 75,25 olur. Bu, `Sales` değerlerinin dağılımında değerlerin %75'inin altına düştüğü noktadır.
`FractileExc (TOTAL Sales, 0.75))`	Tüm `Customer` değerleri için 75,25 olur; çünkü TOTAL niteleyicisinin kullanılması, boyutların göz ardı edilmesi anlamını taşır.
`FractileExc (DISTINCT Sales, 0.75)`	Toplam için 73,50 olur; çünkü DISTINCT niteleyicisinin kullanılması, yalnızca benzersiz `Sales` değerlerinin her bir `Customer` için değerlendirilmesi anlamını taşır.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

Aggr - grafik fonksiyonu (page 410)

Kurtosis

Kurtosis(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadedeki verilerin basıklığını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç
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;	Type, MyKurtosis1 ve MyKurtosis2 boyutlarını içeren bir tabloda, veri kod dosyasındaki Kurtosis() hesaplamaların sonuçları şöyledir: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - grafik fonksiyonu

Kurtosis(), grafik boyutları üzerinde yinelenen ifade veya alanda toplanmış veriler aralığının basıklığını bulur.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

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

İşlev örnekleri

Örnek	Sonuç
`Kurtosis (Value)`	Boyutu ve `Type` `Kurtosis(Value)` hesaplamasını içeren bir tabloyla ilgili olarak, tabloda **Toplamlar** gösterilir ve sayı biçimlendirmesi 3 anlamlı rakama ayarlanırsa sonuç 1,252 olur. `Comparison` için bu değer 1,161 ve `Observation` içinse 1,115 olur.
`Kurtosis (TOTAL Value))`	Tüm `Type` değerleri için 1,252 olur; çünkü TOTAL niteleyicisi boyutların göz ardı edilmesi anlamını taşır.

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

Avg - grafik fonksiyonu (page 270)

LINEST_B

LINEST_B(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplam b değerini (y kesişmesini) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_B - grafik fonksiyonu

LINEST_B(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ifadeleri ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış b değerini (y kesimi) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.

Bağımsız Değişken	Açıklama
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0_const, x0_const	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)
Avg - grafik fonksiyonu (page 270)

LINEST_DF

LINEST_DF(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış serbestlik derecesini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_DF - grafik fonksiyonu

LINEST_DF(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış serbestlik derecelerini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_F

Kod fonksiyonu, bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış F istatistiğini ($r^2/(1-r^2)$) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_F - grafik fonksiyonu

LINEST_F(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış F istatistiğini (r2/(1-r2)) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_M

LINEST_M(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplam m değerini (eğim) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_M - grafik fonksiyonu

LINEST_M(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış m değerini (eğim) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_R2

LINEST_R2(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış r^2 değerini (determinasyon katsayısı) döndürür.

Söz Dizimi:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_R2 - grafik fonksiyonu

LINEST_R2(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış r2 değerini (determinasyon katsayısı) döndürür.

Söz Dizimi:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_SEB

LINEST_SEB(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış b değeri standart hatasını döndürür.

Söz Dizimi:

```
LINEST_SEB (y_değeri, x_değeri[, y0 [, x0 ]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_SEB - grafik fonksiyonu

LINEST_SEB(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait b değeri toplanmış standart hatasını döndürür.

Söz Dizimi:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_SEM

LINEST_SEM(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış m değeri standart hatasını döndürür.

Söz Dizimi:

```
LINEST_SEM (y_değeri, x_değeri[, y0 [, x0 ]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_SEM - grafik fonksiyonu

LINEST_SEM(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait m değeri toplanmış standart hatasını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_SEY

LINEST_SEY(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış y tahmini standart hatasını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_SEY - grafik fonksiyonu

LINEST_SEY(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait y tahmini toplanmış standart hatasını döndürür.

Söz Dizimi:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_SSREG

LINEST_SSREG(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış regresyon kareleri toplamını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_SSREG - grafik fonksiyonu

LINEST_SSREG(), grafik boyutları üzerinde yinelenen, **x_value** ve **y_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış regresyon kareler toplamını döndürür.

Söz Dizimi:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

- *linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)*
- *Avg - grafik fonksiyonu (page 270)*

LINEST_SSRESID

LINEST_SSRESID(), **group by** cümlesi tarafından tanımlandığı şekliyle, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyonun toplanmış kalan kareler toplamını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y(0), x(0)	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

LINEST_SSRESID - grafik fonksiyonu

LINEST_SSRESID(); grafik boyutları üzerinde yinelenen **x_value** ve **y_value** tarafından verilen ifadelerdeki eşleştirilmiş sayılarla temsil edilen bir koordinat dizisi için y=mx+b denklemiyle tanımlanan bir doğrusal regresyona ait toplanmış kalan kareler toplamını döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.
y0, x0	Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir. *Hem y0 hem de x0 belirtilmedikçe fonksiyon, hesaplamak için en az iki geçerli veri çifti gerektirir. y0 ve x0 belirtilirse, tek bir veri çifti yeterli olur.*
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Regresyon çizgisini belirli bir noktada y ekseninden geçmeye zorlayacak şekilde isteğe bağlı bir y0 değeri belirtilebilir. Hem y0 hem de x0 belirtilerek, regresyon çizgisi tek bir sabit koordinattan geçmeye zorlanabilir.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Ayrıca bkz.

linest fonksiyonlarının kullanımına ilişkin örnekler (page 327)

Avg - grafik fonksiyonu (page 270)

Median

Median(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadedeki değerlerin toplanmış medyanını döndürür.

Söz Dizimi:

```
Median (expr)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Örnek: Ortanca kullanan kod ifadesi

Örnek - kod ifadesi

Komut dosyası

Aşağıdaki satır içi veriyi ve kod ifadesini bu örnek için veri yükleme düzenleyicisinde yükleyin.

```
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
```

Görselleştirme oluşturma

Boyutlar olarak **Letter** ve **MyMedian** alanlarını kullanarak bir Qlik Sense sayfasında bir tablo görselleştirmesi oluşturun.

Sonuç

Letter	MyMedian
A	3.5
B	8

Açıklama

Ortanca, sayılar en küçükten en büyüğe sıralandığında "ortada" kalan sayı olarak kabul edilir. Veri kümesinin çift sayıda değeri varsa, fonksiyon ortadaki iki değerin ortalamasını döndürür. Bu örnekte, **A** ve **B** değer setlerinin her biri için, sırasıyla 3,5 ve 8 olan ortanca değerleri hesaplanmaktadır.

Median - grafik fonksiyonu

Median(), grafik boyutları üzerinde yinelenen ifadede toplanmış değerler aralığının medyan değerini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnek: Ortanca kullanan grafik ifadesi

Örnek - grafik ifadesi

Komut dosyası

Aşağıdaki grafik ifadesi örneğini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8 B,9];
```

Görselleştirme oluşturma

Letter alanını boyut olarak kullanarak bir Qlik Sense sayfasında bir tablo görselleştirmesi oluşturun.

Grafik ifadesi

Tabloya hesaplama olarak şu ifadeyi ekleyin:

```
Median(Number)
```

Sonuç

Letter	Median(Number)
Totals	4
A	3.5
B	8

Açıklama

Ortanca, sayılar en küçükten en büyüğe sıralandığında "ortada" kalan sayı olarak kabul edilir. Veri kümesinin çift sayıda değeri varsa, fonksiyon ortadaki iki değerin ortalamasını döndürür. Bu örnekte, **A** ve **B** değer setlerinin her biri için, sırasıyla 3,5 ve 8 olan ortanca değerleri hesaplanmaktadır.

Totals için ortanca, tüm değerlerden hesaplanmaktadır ve 4'e eşittir.

Ayrıca bkz.

Avg - grafik fonksiyonu (page 270)

MutualInfo - grafik fonksiyonu

MutualInfo, iki alan arasındaki veya **Aggr()** içindeki toplu değerler arasındaki karşılıklı bilgileri (MI) hesaplar.

MutualInfo, iki veri kümesi için toplanmış karşılıklı bilgileri döndürür. Bu, bir alan ile potansiyel bir sürücü arasında temel sürücü analizine olanak tanır. Karşılıklı bilgi, veri kümeleri arasındaki ilişkiyi hesaplar ve grafik boyutları üzerinde yinelenen (x,y) çift değerleri için toplanır. Karşılıklı bilgiler 0 ile 1 arasında hesaplanır ve yüzdebirlik değer olarak biçimlendirilebilir. **MutualInfo** ya seçimlerle ya da bir küme ifadesi ile tanımlanır.

MutualInfo farklı türden MI analizlerine izin verir:

- İkili MI: Sürücü alanı ile hedef alan arasındaki MI'yı hesaplayın.
- Değere göre sürücü kırılımı: MI, sürücü ve hedef alanlarındaki bireysel alan değerleri arasında hesaplanır.
- Özellik seçimi: Tüm alanların MI'ye göre birbiriyle karşılaştırıldığı bir matris oluşturmak için ızgara grafikte **MutualInfo** kullanın.

MutualInfo mutlaka karşılıklı bilgi paylaşan alanlar arasında nedensellik göstermez. İki alan karşılıklı bilgi paylaşabilir, ancak birbirleri için eşit sürücü olmayabilir. Örneğin, dondurma satışları ile dışarıdaki hava sıcaklığını karşılaştırırken, **MutualInfo** ikisi arasında karşılıklı bilgi gösterecektir. Dondurma satışlarını artıranın dış hava sıcaklığı mı olduğunu (bu mümkündür), dış hava sıcaklığını artıranın dondurma satışları mı olduğunu (bu mümkün değildir) göstermez.

Karşılıklı bilgi hesaplanırken, ilişkilendirmeler, farklı tablolardan gelen alanlardaki değerlerin sıklığını ve aralarındaki ilişkiyi etkiler.

Aynı alanlar veya seçimler için döndürülen değerler biraz farklılık gösterebilir. Bunun nedeni, her **MutualInfo** çağrısının rastgele seçilmiş bir örnek üzerinde çalışması ve **MutualInfo** algoritmasının doğal rastgeleliğidir.

MutualInfo, **Aggr()** fonksiyonuna uygulanabilir.

Söz Dizimi:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [, breakdownbyvalue [, samplesize ]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field1, field2	Karşılıklı bilgilerin hesaplanacağı iki örnek kümeyi içeren ifadeler veya alanlar.
datatype	Hedef ve sürücüdeki veri türleri, 1 veya discrete:discrete için 'dd' 2 veya continuous:continuous için 'cc' 3 veya continuous:discrete için 'cd' 4 veya discrete:continuous için 'dc' Veri türleri küçük/büyük harfe duyarlı değildir.
breakdownbyvalue	Sürücüdeki bir değere karşılık gelen statik değer. Sağlanmışsa, hesaplama bu değer için MI katkısını hesaplayacaktır. **ValueList()** veya **ValueLoop()** kullanabilirsiniz. **Null()** eklenirse, hesaplama sürücüdeki tüm değerler için genel MI değerini hesaplayacaktır. Değere göre döküm, sürücünün ayrık veriler içermesini gerektirir.
samplesize	Hedef ve sürücüden örnek oluşturulacak değerlerin sayısı. Örnek oluşturma rastgeledir. **MutualInfo** örnek boyutunun en az 80 olmasını gerektirir. **MutualInfo** kaynakları yoğun şekilde kullanabileceği için **MutualInfo** varsayılan olarak sadece 10.000 diziye kadar örnekleme yapar. Örnek boyutunda, daha fazla sayıda veri çifti belirtebilirsiniz. **MutualInfo** zaman aşımına uğrarsa örnek boyutunu azaltın.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Fonksiyon örnekleri

Örnek	Sonuç
`mutualinfo(Age, Salary, 1)`	`Employee name` boyutunu ve `mutualinfo(Age, Salary, 1)` hesaplamasını içeren bir tablo için sonuç 0.99820986 olur. Sonuç yalnızca toplamlar hücresi için görüntülenir.
`mutualinfo (TOTAL Age, Salary, 1, null (), 81)`	Gender boyutuyla bir filtre bölmesi oluşturursanız ve bundan seçimler yaparsanız, Female seçildiğinde 0,99805677, Male seçildiğinde 0,99847373 sonucunu görürsünüz. Bunun nedeni, seçimin diğer Gender değerine ait olmayan tüm sonuçları hariç tutmasıdır.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Gender üzerinde herhangi bir değer seçmek, bunu 0 olarak değiştirecektir.
`mutualinfo({1} TOTAL Age, Salary, 1, null ())`	0.99820986. Bu, seçimlerden bağımsızdır. Bunun nedeni, {1} küme ifadesinin tüm seçimleri ve boyutları göz ardı etmesidir.

Örneklerde kullanılan veriler:

```
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(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifadenin eğriliğini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
DISTINCT	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyutlar olarak `Type` ve *MySkew* ile düz tablo oluşturun.

Sonuç verileri

Örnek	Sonuç
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;	Skew() hesaplamasının sonuçları: • Type : MySkew • Comparison : 0.86414768 • Observation : 0.32625351

Skew - grafik fonksiyonu

Skew(), grafik boyutları üzerinde yinelenen ifadenin veya alanın toplanmış eğriliğini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {,fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyut olarak `Type` ve hesaplama olarak `Skew (Value)` ile düz tablo oluşturun.

Tablonun özelliklerinde `Totals` etkinleştirilmelidir.

Örnek	Sonuç

`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`
`'|');`
	Skew(Value) hesaplamasının sonuçları:
	• `Total`: 0.23522195
	• `Comparison` : 0.86414768
	• `Observation` : 0.32625351

Ayrıca bkz.

Avg - grafik fonksiyonu (page 270)

Stdev

Stdev(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde, ifade ile verilen değerlerin standart sapmasını döndürür.

Söz Dizimi:

```
Stdev([distinct] expr)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyutlar olarak `Type` ve `MyStdev` ile düz tablo oluşturun.

Sonuç verileri

Örnek	Sonuç
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;	Stdev() hesaplamasının sonuçları: • Type : MyStdev • Comparison : 14.61245 • Observation : 12.507997

Stdev - grafik fonksiyonu

Stdev(), grafik boyutları üzerinde yinelenen ifade veya alanda toplanmış veriler aralığının standart sapmasını bulur.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyut olarak `Type` ve hesaplama olarak `Stdev (Value)` ile düz tablo oluşturun.

Tablonun özelliklerinde `Totals` etkinleştirilmelidir.

Örnek	Sonuç

`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`
`'|');`
	Stdev(Value) hesaplamasının sonuçları:
	• `Total`: 15.47529
	• `Comparison` : 14.61245
	• `Observation` : 12.507997

Ayrıca bkz.

- *Avg - grafik fonksiyonu (page 270)*
- *STEYX - grafik fonksiyonu (page 325)*

Sterr

Sterr(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen ifadenin temsil ettiği bir değerler dizisi için toplanmış standart hatayı (stdev/sqrt(n)) döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
distinct	İfadeden önce **distinct** sözcüğü varsa, tüm çoğaltmalar göz ardı edilir.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç

`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;`
	`Type` ve `MySterr` boyutlarını içeren bir tabloda, veri kod dosyasındaki Sterr() hesaplamanın sonuçları şöyledir:
	
	`Type MySterr`
	`Comparison 3.2674431`
	`Observation 2.7968733`

Sterr - grafik fonksiyonu

Sterr(), grafik boyutları üzerinde yinelenen ifadede toplanmış değer dizisi için ortalamanın standart hatası değerini (stdev/sqrt(n)) bulur.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyut olarak `Type` ve hesaplama olarak `Sterr (Value)` ile düz tablo oluşturun.

Tablonun özelliklerinde `Totals` etkinleştirilmelidir.

Örnek	Sonuç
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 '|');	Sterr(Value) hesaplamasının sonuçları: • Total: 2.4468583 • Comparison : 3.2674431 • Observation : 2.7968733

Ayrıca bkz.

Avg - grafik fonksiyonu (page 270)
STEYX - grafik fonksiyonu (page 325)

STEYX

STEYX(), bir **group by** cümlesi ile tanımlandığı şekilde, bir dizi kayıt üzerinde yinelenen x-expression ve y-expression içindeki eşlenmiş sayılarca temsil edilen bir koordinatlar dizisi için regresyondaki her bir x değeri için tahmini y değerinin toplam standart hatasını döndürür.

Söz Dizimi:

```
STEYX (y_value, x_value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak x değerleri aralığını içeren ifade veya alan.

Sınırlamalar:

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç

`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;`
	`MySTEYX` boyutunu içeren bir tabloda, veri kod dosyasındaki STEYX() hesaplamasının sonucu 2,0714764 olur.

STEYX - grafik fonksiyonu

STEYX(), **y_value** ve **x_value** ile verilen ifadelerdeki eşleştirilmiş sayıların temsil ettiği bir koordinat dizisi ile verilen doğrusal regresyonda her bir x değeri için y değerlerini tahmin ederken toplanmış standart hatayı döndürür.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
y_value	Hesaplanacak bilinen y değerleri aralığını içeren ifade veya alan.
x_value	Hesaplanacak bilinen x değerleri aralığını içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

Bu iç toplamalar **TOTAL** niteleyicisini içermedikçe, toplama işlevinin parametresi diğer toplama işlevlerini içermemelidir. Daha gelişmiş iç içe toplamalar için, belirli bir boyutla birlikte gelişmiş **Aggr** işlevini kullanın.

Bir veri çiftinin herhangi bir veya her iki parçasındaki metin değerleri, NULL değerler ve eksik değerler, veri çiftinin tamamının göz ardı edilmesine neden olur.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Ardından boyut olarak `KnownY` ve `KnownX` ve hesaplama olarak `Steyx(KnownY,KnownX)` ile düz tablo oluşturun.

Tablonun özelliklerinde `Totals` etkinleştirilmelidir.

Örnek	Sonuç
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 '|');	STEYX(KnownY,KnownX) hesaplamasının sonucu 2,071'dir (Sayı biçimlendirmesi 3 ondalık basamağa ayarlanırsa.)

Ayrıca bkz.

- *Avg - grafik fonksiyonu (page 270)*
- *Sterr - grafik fonksiyonu (page 322)*

linest fonksiyonlarının kullanımına ilişkin örnekler

linest fonksiyonları, doğrusal regresyon analizi ile ilişkili değerleri bulmak için kullanılır. Bu bölümde, Qlik Sense içinde kullanılabilen linest fonksiyonlarının değerlerini bulmak için örnek veriler kullanılarak görselleştirmelerin nasıl oluşturulacağı açıklanmaktadır. linest fonksiyonları veri kod dosyasında ve grafik ifadelerinde kullanılabilir.

Söz dizimi ve bağımsız değişkenler ile ilgili açıklamalar için lütfen, ayrı linest grafik fonksiyonu ve kod fonksiyonu konularına bakın.

Örneklerde kullanılan veri ve kod ifadeleri

Şu satır içi veri ve kod ifadelerini aşağıdaki linest() örnekleri için veri yükleme düzenleyicisine yükleyin.

```
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;
```

Örnek 1: linest kullanan kod ifadeleri

Örnek: Kod ifadeleri

Veri yükleme kod hesaplamalarından bir görselleştirme oluşturun

Şu alanları sütun olarak kullanarak bir Qlik Sense sayfasında bir tablo görselleştirmesi oluşturun:

- Linest_B
- Linest_DF

- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Sonuç

Veri kod dosyasında yapılan linest hesaplamalarının sonuçlarını içeren tablo şöyle görünmelidir:

Sonuçlar tablosu

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

Sonuçlar tablosu

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Örnek 2: linest kullanan grafik ifadeleri

Örnek: Grafik ifadeleri

Şu alanları boyut olarak kullanarak bir Qlik Sense sayfasında bir görselleştirme oluşturun:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Bu ifadede linest fonksiyonlarının adlarıyla boyutlara ilişkin etiketler oluşturmak için yapay boyutlar fonksiyonu kullanılmaktadır. Yerden kazanmak için etiketi **Linest functions** olarak değiştirebilirsiniz.

Tabloya hesaplama olarak şu ifadeyi ekleyin:

```
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) )
```

Bu ifade, her bir linest fonksiyonunun sonuç değerini, yapay boyuttaki ilgili ada karşı görüntüler. `Linest_b(Y,X)` sonucu, **linest_b** öğesinin yanında görüntülenir ve bu böyle devam eder.

Sonuç

Sonuçlar tablosu

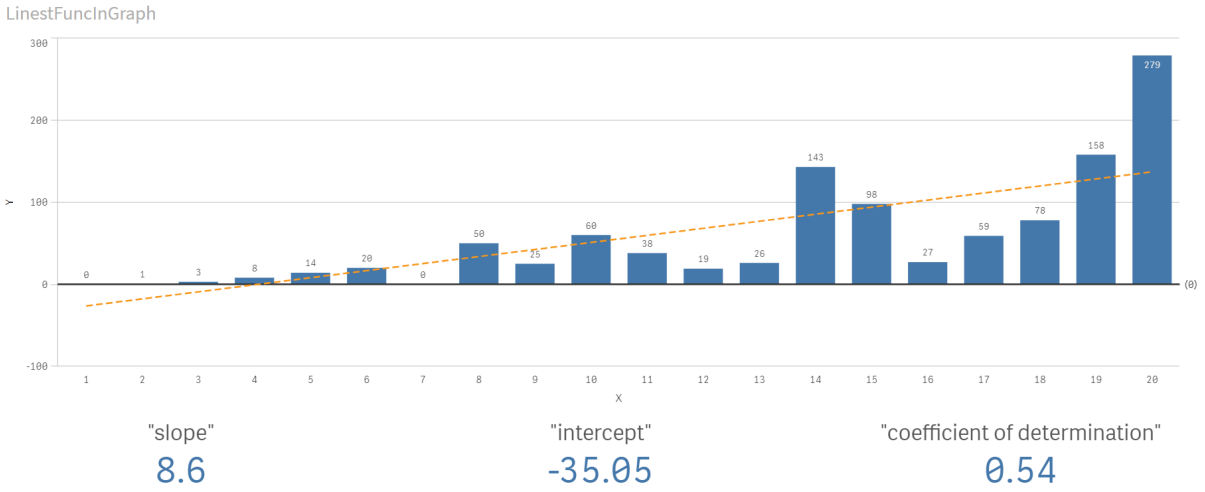
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Örnek 3: linest kullanan grafik ifadeleri

Örnek: Grafik ifadeleri

1. Bir Qlik Sense sayfasında, **X** değerini boyut ve **Y** değerini hesaplama için kullanarak bir çubuk grafik görselleştirmesi oluştur.
2. Y hesaplamasına doğrusal bir eğilim çizgisi ekle.
3. Sayfaya bir KPI görselleştirmesi ekle.
    1. *Eğimi* KPI için bir etiket olarak ekle.
    2. `sum(Linest_M)` öğesini KPI için bir ifade olarak ekle.
4. Sayfaya ikinci bir KPI görselleştirmesi ekle.
    1. *İntersepti* KPI için bir etiket olarak ekle.
    2. `Sum(Linest_B)` öğesini KPI için bir ifade olarak ekle.
5. Sayfaya üçüncü bir KPI görselleştirmesi ekle.
    1. *Determinasyon katsayısını* KPI için bir etiket olarak ekle.
    2. `Sum(Linest_R2)` öğesini KPI için bir ifade olarak ekle.

Sonuç



"slope"	"intercept"	"coefficient of determination"
8.6	-35.05	0.54

Açıklama

Çubuk grafiği, X ve Y verilerinin çizilmesini göstermektedir. İlgili linest() fonksiyonları, eğilim çizgisinin temel aldığı doğrusal regresyon denklemi; yani y = m * x + b için değerler sağlar. Denklem, veriye en iyi uyan çizgiyi betimleyen bir dizi döndürerek düz bir çizgi (eğilim çizgisi) hesaplamak için "en düşük kareler" yöntemini kullanır.

KPI'lar; doğrusal regresyon denkleminde değişkenler olan eğim için **sum(Linest_M)** ve Y intersepti için **sum(Linest_B)** linest() fonksiyonlarının sonuçlarını ve determinasyon katsayısı için ilgili toplanmış R2 değerini görüntüler.

İstatistiksel test fonksiyonları

İstatistiksel test işlevleri, hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir, ancak sözdizimi farklıdır.

Ki2 testi fonksiyonları

Genellikle nitel değişkenlerin incelenmesinde kullanılır. Gözlenen sıklıklar beklenen sıklıkları içeren tek yönlü bir sıklık tablosunda karşılaştırılabilir veya iki değişken arasındaki bağlantı bir olumsallık (kontenjan) tablosunda incelenebilir.

T testi fonksiyonları

T testi fonksiyonları iki popülasyon ortalamasının istatistiksel incelemesi için kullanılır. İki örnekli t testi iki örneğin farklı olup olmadığını inceler; iki normal dağılımın bilinmeyen varyanslara sahip olduğu ve deneyde küçük örneklem boyutu kullanıldığı durumlarda yaygın olarak kullanılır.

Z testi fonksiyonları

İki popülasyon ortalamasının istatistiksel incelemesi. İki örnek z testi, iki normal dağıtımın bilinen varyansları olduğunda ve bir deneme büyük bir örnek boyutu kullandığında iki örneğin farklı olup olmadığını ve sık kullanılıp kullanılmadığını inceler.

Ki2 testi fonksiyonları

Genellikle nitel değişkenlerin incelenmesinde kullanılır. Gözlenen sıklıklar beklenen sıklıkları içeren tek yönlü bir sıklık tablosunda karşılaştırılabilir veya iki değişken arasındaki bağlantı bir olumsallık (kontenjan) tablosunda incelenebilir.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.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Chi2Test_chi2
Chi2Test_chi2(), bir veya iki değer dizisi için toplanmış ki^2 testi değerini döndürür.

```
Chi2Test_chi2(), bir veya iki değer dizisi için toplanmış ki2 testi değerini
döndürür.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df(), bir veya iki değer dizisi için toplanmış ki^2 testi df değerini (serbestlik derecesi) döndürür.

```
Chi2Test_df(), bir veya iki değer dizisi için toplanmış ki2 testi df değerini
(serbestlik derecesi) döndürür.(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p(), bir veya iki değer dizisi için toplanmış ki^2 testi p değerini (anlamlılık) döndürür.

```
Chi2Test_p - grafik fonksiyonu(col, row, actual_value[, expected_value])
```

Ayrıca bkz.

- *T testi fonksiyonları (page 334)*
- *Z testi fonksiyonları (page 368)*

Chi2Test_chi2

Chi2Test_chi2(), bir veya iki değer dizisi için toplanmış ki^2 testi değerini döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Tüm Qlik Sense chi^2 testi fonksiyonları aynı bağımsız değişkenlere sahiptir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
col, row	Test edilmekte olan değerlerin matrisinde belirtilen sütun ve satır.
actual_value	Belirtilen **col** ve **row** için verilerin gözlenen değeri.
expected_value	Belirtilen **col** ve **row** için beklenen dağılım değeri.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Ayrıca bkz.

- *Grafiklerde chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 384)*
- *Veri yükleme komut dosyasında chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 387)*

Chi2Test_df

Chi2Test_df(), bir veya iki değer dizisi için toplanmış ki^2 testi df değerini (serbestlik derecesi) döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Tüm Qlik Sense chi^2 testi fonksiyonları aynı bağımsız değişkenlere sahiptir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
col, row	Test edilmekte olan değerlerin matrisinde belirtilen sütun ve satır.
actual_value	Belirtilen **col** ve **row** için verilerin gözlenen değeri.
expected_value	Belirtilen **col** ve **row** için beklenen dağılım değeri.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Ayrıca bkz.

Grafiklerde chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 384)
Veri yükleme komut dosyasında chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 387)

Chi2Test_p - grafik fonksiyonu

Chi2Test_p(), bir veya iki değer dizisi için toplanmış ki^2 testi p değerini (anlamlılık) döndürür. Test, belirtilen **col** ve **row** matrisi dahilindeki değişiklikleri test edecek şekilde **actual_value** içindeki değerler üzerinde veya **actual_value** içindeki değerleri **expected_value** içindeki karşılık gelen değerlerle karşılaştırarak (belirtilirse) yapılabilir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Tüm Qlik Sense chi^2 testi fonksiyonları aynı bağımsız değişkenlere sahiptir.

Söz Dizimi:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
col, row	Test edilmekte olan değerlerin matrisinde belirtilen sütun ve satır.
actual_value	Belirtilen **col** ve **row** için verilerin gözlenen değeri.
expected_value	Belirtilen **col** ve **row** için beklenen dağılım değeri.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Ayrıca bkz.

Grafiklerde chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 384)
Veri yükleme komut dosyasında chi2-test fonksiyonlarının kullanımına ilişkin örnekler (page 387)

T testi fonksiyonları

T testi fonksiyonları iki popülasyon ortalamasının istatistiksel incelemesi için kullanılır. İki örnekli t testi iki örneğin farklı olup olmadığını inceler; iki normal dağılımın bilinmeyen varyanslara sahip olduğu ve deneyde küçük örneklem boyutu kullanıldığı durumlarda yaygın olarak kullanılır.

Aşağıdaki bölümlerde, t testi istatistiksel test fonksiyonları, her bir fonksiyon türüne uygulanan örnek öğrenci testine göre gruplandırılmıştır.

Tipik bir t-test raporu oluşturma (page 389)

İki bağımsız örnek t testleri

Aşağıdaki fonksiyonlar, iki bağımsız örnek öğrenci t testi için geçerlidir.

ttest_conf
TTest_conf, iki bağımsız örnek için toplanmış t testi güven aralığı değerini döndürür.

```
TTest_conf, iki bağımsız örnek için toplanmış t testi güven aralığı değerini
döndürür. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi değerini (serbestlik derecesi) döndürür.

```
TTest_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi değerini
(serbestlik derecesi) döndürür. (grp, value [, eq_var)
```

ttest_dif

TTest_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama farkını döndüren bir sayısal fonksiyondur.

```
TTest_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi
ortalama farkını döndüren bir sayısal fonksiyondur. (grp, value)
```

ttest_lower

TTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

```
TTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna
yönelik toplanmış değeri döndürür. (grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2 kuyruklu anlamlılık düzeyini döndürür.

```
TTest_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2
kuyruklu anlamlılık düzeyini döndürür. (grp, value [, eq_var])
```

ttest_sterr

TTest_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

```
TTest_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi
ortalama fark standart hatasını döndürür. (grp, value [, eq_var])
```

ttest_t

TTest_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

```
TTest_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.
(grp, value [, eq_var])
```

ttest_upper

TTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

```
TTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna
yönelik toplanmış değeri döndürür. (grp, value [, sig [, eq_var]])
```

İki bağımsız ağırlıklı örnek t testleri

Aşağıdaki fonksiyonlar, giriş veri serisinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek Student t testi için geçerlidir.

ttestw_conf

TTestw_conf(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

TTestw_conf(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.
(weight, grp, value [, sig[, eq_var]]**)**

ttestw_df

TTestw_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

TTestw_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür. (weight, grp, value [, eq_var]**)**

ttestw_dif

TTestw_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama farkını döndürür.

TTestw_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama farkını döndürür. (weight, grp, value**)**

ttestw_lower

TTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

TTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_sig

TTestw_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2 kuyruklu anlamlılık düzeyini döndürür.

TTestw_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2 kuyruklu anlamlılık düzeyini döndürür. (weight, grp, value [, eq_var]**)**

ttestw_sterr

TTestw_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

TTestw_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür. (weight, grp, value [, eq_var]**)**

ttestw_t

TTestw_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

TTestw_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.
(weight, grp, value [, eq_var]**)**

ttestw_upper

TTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

TTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür. (weight, grp, value [, sig [, eq_var]]**)**

Tek örnek t testleri

Aşağıdaki fonksiyonlar, tek örnekli Student t testi için geçerlidir.

ttest1_conf

TTest1_conf(), bir değer dizisi için toplanmış güven aralığı değerini döndürür.

```
TTest1_conf(), bir değer dizisi için toplanmış güven aralığı değerini
döndürür. (value [, sig])
```

ttest1_df

TTest1_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

```
TTest1_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini
(serbestlik derecesi) döndürür. (value)
```

ttest1_dif

TTest1_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi ortalaması farkını döndürür.

```
TTest1_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi
ortalaması farkını döndürür. (value)
```

ttest1_lower

TTest1_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

```
TTest1_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik
toplanmış değeri döndürür. (value [, sig])
```

ttest1_sig

TTest1_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t testi 2 kuyruklu belirgin düzeyini döndürür.

```
TTest1_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t testi
2 kuyruklu belirgin düzeyini döndürür. (value)
```

ttest1_sterr

TTest1_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

```
TTest1_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama fark
standart hatasını döndürür. (value)
```

ttest1_t

TTest1_t() bir değer dizisi için toplanmış t değerini döndürür.

```
TTest1_t() bir değer dizisi için toplanmış t değerini döndürür. (value)
```

ttest1_upper

TTest1_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

```
TTest1_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik
toplanmış değeri döndürür. (value [, sig])
```

Tek ağırlıklı örnek t testleri

Aşağıdaki fonksiyonlar giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli Student t testi için geçerlidir.

ttest1w_conf

TTest1w_conf(), bir değer dizisi için toplanmış güven aralığı değerini döndüren bir **sayısal** fonksiyondur.

```
TTest1w_conf(), bir değer dizisi için toplanmış güven aralığı değerini
döndüren bir sayısal fonksiyondur. (weight, value [, sig])
```

ttest1w_df

TTest1w_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

```
TTest1w_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini
(serbestlik derecesi) döndürür. (weight, value)
```

ttest1w_dif

TTest1w_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi ortalaması farkını döndürür.

```
TTest1w_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi
ortalaması farkını döndürür. (weight, value)
```

ttest1w_lower

TTest1w_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

```
TTest1w_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik
toplanmış değeri döndürür. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t testi 2 kuyruklu belirgin düzeyini döndürür.

```
TTest1w_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t
testi 2 kuyruklu belirgin düzeyini döndürür. (weight, value)
```

ttest1w_sterr

TTest1w_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

```
TTest1w_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama
fark standart hatasını döndürür. (weight, value)
```

ttest1w_t

TTest1w_t() bir değer dizisi için toplanmış t değerini döndürür.

```
TTest1w_t() bir değer dizisi için toplanmış t değerini döndürür. ( weight,
value)
```

ttest1w_upper

TTest1w_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

```
TTest1w_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik
toplanmış değeri döndürür. (weight, value [, sig])
```

TTest_conf

TTest_conf, iki bağımsız örnek için toplanmış t testi güven aralığı değerini döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_df

TTest_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi değerini (serbestlik derecesi) döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_dif

TTest_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama farkını döndüren bir sayısal fonksiyondur.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_lower

TTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_sig

TTest_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2 kuyruklu anlamlılık düzeyini döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_sterr

TTest_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_t

TTest_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest_t( Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest_upper

TTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_conf

TTestw_conf(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_df

TTestw_df(), iki bağımsız değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_dif

TTestw_dif(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama farkını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_lower

TTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_sig

TTestw_sig(), iki bağımsız değer dizisi için toplanmış öğrenci t testi 2 kuyruklu anlamlılık düzeyini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_sterr

TTestw_sterr(), iki bağımsız değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_t

TTestw_t(), iki bağımsız değer dizisi için toplanmış t değerini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTestw_upper

TTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği iki bağımsız örnek öğrenci t testleri içindir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_conf

TTest1_conf(), bir değer dizisi için toplanmış güven aralığı değerini döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_df

TTest1_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_df (value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1_df( Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_dif

TTest1_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi ortalaması farkını döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_dif (value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1_dif( Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_lower

TTest1_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_lower (value [, sig])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_sig

TTest1_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t testi 2 kuyruklu belirgin düzeyini döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_sig (value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1_sig( Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_sterr

TTest1_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_sterr (value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1_sterr( Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_t

TTest1_t() bir değer dizisi için toplanmış t değerini döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_t (value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1_t( Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1_upper

TTest1_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, tek örnekli öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1_upper (value [, sig])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.

Bağımsız Değişken	Açıklama
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_conf

TTest1w_conf(), bir değer dizisi için toplanmış güven aralığı değerini döndüren bir **sayısal** fonksiyondur.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1w_conf (weight, value [, sig ])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_df

TTest1w_df(), bir değer dizisi için toplanmış öğrenci t testi df değerini (serbestlik derecesi) döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1w_df (weight, value)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1w_df( Weight, Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_dif

TTest1w_dif(), bir değer dizisi için birleştirilmiş öğrencinin t testi ortalaması farkını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1w_dif( Weight, Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_lower

TTest1w_lower(), bir değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_sig

TTest1w_sig(), bir değer dizisi için anlamlı değerin toplanmış öğrenci t testi 2 kuyruklu belirgin düzeyini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1w_sig( Weight, Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_sterr

TTest1w_sterr(), bir değer dizisi için toplanmış öğrenci t testi ortalama fark standart hatasını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1w_sterr( Weight, Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_t

TTest1w_t() bir değer dizisi için toplanmış t değerini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
TTest1w_t( Weight, Value )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

TTest1w_upper

TTest1w_upper(), bir değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği tek örnekli öğrenci t testleri için geçerlidir:

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
TTest1w_upper (weight, value [, sig])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnekler. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
weight	**value** içindeki her bir değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Ayrıca bkz.

Tipik bir t-test raporu oluşturma (page 389)

Z testi fonksiyonları

İki popülasyon ortalamasının istatistiksel incelemesi. İki örnek z testi, iki normal dağıtımın bilinen varyansları olduğunda ve bir deneme büyük bir örnek boyutu kullandığında iki örneğin farklı olup olmadığını ve sık kullanılıp kullanılmadığını inceler.

Z testi istatistiksel test fonksiyonları, fonksiyona uygulanan giriş veri serilerinin türüne göre gruplandırılır.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

Tek sütun biçiminde fonksiyonlar

Aşağıdaki fonksiyonlar, basit giriş veri serilerini içeren z testleri için geçerlidir.

ztest_conf

ZTest_conf(), bir değer dizisi için toplanmış z değerini döndürür.

```
ZTest_conf(), bir değer dizisi için toplanmış z değerini döndürür. (value [,
sigma [, sig ])
```

ztest_dif

ZTest_dif(), bir değer dizisi için toplanmış z testi ortalama farkını döndürür.

```
ZTest_dif(), bir değer dizisi için toplanmış z testi ortalama farkını
döndürür. (value [, sigma])
```

ztest_sig

ZTest_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık düzeyini döndürür.

```
ZTest_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık
düzeyini döndürür. (value [, sigma])
```

ztest_sterr

ZTest_sterr(), bir değer dizisi için toplanmış z testi ortalama fark standart hatasını döndürür.

```
ZTest_sterr(), bir değer dizisi için toplanmış z testi ortalama fark standart
hatasını döndürür. (value [, sigma])
```

ztest_z

ZTest_z(), bir değer dizisi için toplanmış z değerini döndürür.

```
ZTest_z(), bir değer dizisi için toplanmış z değerini döndürür. (value [,
sigma])
```

ztest_lower

ZTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

```
ZTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna
yönelik toplanmış değeri döndürür. (grp, value [, sig [, eq_var]])
```

ztest_upper

ZTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

```
ZTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna
yönelik toplanmış değeri döndürür. (grp, value [, sig [, eq_var]])
```

Ağırlıklı iki sütun biçiminde fonksiyonlar

Aşağıdaki fonksiyonlar, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

ztestw_conf

ZTestw_conf(), bir değer dizisi için toplanmış z güven aralığı değerini döndürür.

```
ZTestw_conf(), bir değer dizisi için toplanmış z güven aralığı değerini
döndürür. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif(), bir değer dizisi için toplanmış z testi ortalama farkını döndürür.

```
ZTestw_dif(), bir değer dizisi için toplanmış z testi ortalama farkını
döndürür.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

```
ZTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna
yönelik toplanmış değeri döndürür. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık düzeyini döndürür.

```
ZTestw_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık
düzeyini döndürür. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr(), bir değer dizisi için toplanmış z testi ortalama fark standart hatasını döndürür.

```
ZTestw_sterr(), bir değer dizisi için toplanmış z testi ortalama fark
standart hatasını döndürür. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

```
ZTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna
yönelik toplanmış değeri döndürür. (weight, value [, sigma])
```

ztestw_z

ZTestw_z(), bir değer dizisi için toplanmış z değerini döndürür.

```
ZTestw_z(), bir değer dizisi için toplanmış z değerini döndürür. (weight,
value [, sigma])
```

ZTest_z

ZTest_z(), bir değer dizisi için toplanmış z değerini döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_z(value[, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTest_z( Value-TestValue )
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_sig

ZTest_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık düzeyini döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_sig(value[, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTest_sig(Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_dif

ZTest_dif(), bir değer dizisi için toplanmış z testi ortalama farkını döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_dif(value[, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTest_dif(Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_sterr

ZTest_sterr(), bir değer dizisi için toplanmış z testi ortalama fark standart hatasını döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_sterr(value[, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTest_sterr(Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_conf

ZTest_conf(), bir değer dizisi için toplanmış z değerini döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_conf(value[, sigma[, sig]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTest_conf(Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_lower

ZTest_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTest_upper

ZTest_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_z

ZTestw_z(), bir değer dizisi için toplanmış z değerini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTestw_z (weight, value [, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerler **value** tarafından döndürülmelidir. 0 örnek ortalaması kabul edilir. Testin başka bir ortalama çevresinde gerçekleştirilmesini istiyorsanız, örnek değerlerden söz konusu değeri çıkartın.
weight	**value** içindeki her bir örnek değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTestw_z( Weight, Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_sig

ZTestw_sig(), bir değer dizisi için toplanmış z testi 2 kuyruklu anlamlılık düzeyini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTestw_sig (weight, value [, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerler **value** tarafından döndürülmelidir. 0 örnek ortalaması kabul edilir. Testin başka bir ortalama çevresinde gerçekleştirilmesini istiyorsanız, örnek değerlerden söz konusu değeri çıkartın.
weight	**value** içindeki her bir örnek değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTestw_sig( Weight, Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_dif

ZTestw_dif(), bir değer dizisi için toplanmış z testi ortalama farkını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTestw_dif ( weight, value [, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerler **value** tarafından döndürülmelidir. 0 örnek ortalaması kabul edilir. Testin başka bir ortalama çevresinde gerçekleştirilmesini istiyorsanız, örnek değerlerden söz konusu değeri çıkartın.
weight	**value** içindeki her bir örnek değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTestw_dif( Weight, Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_sterr

ZTestw_sterr(), bir değer dizisi için toplanmış z testi ortalama fark standart hatasını döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTestw_sterr (weight, value [, sigma])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerler **value** tarafından döndürülmelidir. 0 örnek ortalaması kabul edilir. Testin başka bir ortalama çevresinde gerçekleştirilmesini istiyorsanız, örnek değerlerden söz konusu değeri çıkartın.
weight	**value** içindeki her bir örnek değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_conf

ZTestw_conf(), bir değer dizisi için toplanmış z güven aralığı değerini döndürür.

Bu fonksiyon, giriş veri serilerinin ağırlıklı iki sütun biçiminde verildiği z testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Popülasyon ortalamasının 0 olduğu varsayılır. Testin başka bir ortalama etrafında gerçekleştirilmesini istiyorsanız, örnek değerlerden o ortalamayı çıkarın.
weight	**value** içindeki her bir örnek değer, **weight** içindeki karşılık gelen ağırlık değerine göre bir veya daha fazla kez sayılabilir.
sigma	Standart sapma biliniyorsa, **sigma** içinde belirtilebilir. **sigma** atlanırsa, gerçek örnek standart sapması kullanılır.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnek:

```
ZTestw_conf( Weight, Value-TestValue)
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_lower

ZTestw_lower(), iki bağımsız değer dizisi için güven aralığının alt ucuna yönelik toplanmış değeri döndürür.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

ZTestw_upper

ZTestw_upper(), iki bağımsız değer dizisi için güven aralığının üst ucuna yönelik toplanmış değeri döndürür.

Bu fonksiyon, bağımsız örnekler öğrenci t testleri için geçerlidir.

Fonksiyon veri kod dosyasında kullanılırsa, değerler group by cümlesi tarafından tanımlandığı şekliyle bir dizi kayıt üzerinden tekrarlanır.

Fonksiyon bir grafik ifadesinde kullanılırsa, değerler grafik boyutları üzerinde yinelenir.

Söz Dizimi:

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

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Değerlendirilecek örnek değerler. Örnek değerler, **group** içinde tam olarak iki değer ile belirtildiği şekilde mantıksal olarak gruplandırılmalıdır. Örnek değerler için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Value** adı verilir.
grp	İki örnek grubun her birinin adlarını içeren alan. Grup için bir alan adı kod dosyasında sağlanmazsa, alana otomatik olarak **Type** adı verilir.
sig	Anlamlı değerin iki kuyruklu düzeyi **sig** içinde belirtilebilir. Atlanırsa, **sig** 0,025 olarak ayarlanır ve bu da %95 oranında bir güven aralığıyla sonuçlanır.
eq_var	**eq_var** değeri False (0) olarak belirtilirse, iki örneğin ayrı varyansları olduğu varsayılır. **eq_var** değeri True (1) olarak belirtilirse, örnekler arasında eşit varyanslar olduğu varsayılır.

Sınırlamalar:

İfade değerindeki metin değerleri, NULL değerler ve eksik değerler fonksiyonun NULL sonucunu döndürmesiyle sonuçlanır.

Örnekler:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Ayrıca bkz.

z-test fonksiyonlarının kullanımına ilişkin örnekler (page 392)

İstatistiksel test fonksiyonu örnekleri

Bu bölümde, grafiklere ve veri kod dosyasına uygulandığı şekliyle istatistiksel test fonksiyonlarının örnekleri yer almaktadır.

Grafiklerde chi2-test fonksiyonlarının kullanımına ilişkin örnekler

chi2-test fonksiyonları, ki-kareli istatistiksel analiz ile ilişkili değerleri bulmak için kullanılır.

Bu bölümde, Qlik Sense içinde kullanılabilen ki-kareli dağılım test fonksiyonlarının değerlerini bulmak için örnek veriler kullanılarak görselleştirmelerin nasıl oluşturulacağı açıklanmaktadır. Söz dizimi ve bağımsız değişkenler ile ilgili açıklamalar için lütfen, ayrı chi2-test grafik fonksiyonu konularına bakın.

Örnekler için verileri yükleme

Koda yüklenecek üç farklı istatistiksel örneklemi açıklayan üç örnek veri kümesi vardır.

Aşağıdakileri yapın:

1. Yeni bir uygulama oluşturun.
2. Veri yüklemesine aşağıdakileri girin:

```
// 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. Verileri yüklemek için seçeneğine tıklayın.

chi2-test grafik fonksiyonu görselleştirmelerini oluşturma

Örnek: Örnek 1

Aşağıdakileri yapın:

1. Veri yükleme düzenleyicisinde, seçeneğine tıklayarak uygulama görünümüne gidin ve daha önce oluşturduğunuz sayfaya tıklayın.
   Sayfa görünümü açılır.
2. Sayfayı düzenlemek için **Sayfayı düzenle** seçeneğine tıklayın.
3. **Grafikler**'den bir tablo ekleyin ve **Alanlar**'dan boyutlar olarak Grp, Grade ve Count ekleyin.
   Bu tabloda örnek veriler gösterilmektedir.
4. Boyut olarak aşağıdaki ifadeyle başka bir tablo ekleyin:
   `ValueList('p','df','Chi2')`
   Böylece, üç chi2-test fonksiyonunun adlarıyla boyutlara ilişkin etiketler oluşturmak için yapay boyutlar fonksiyonu kullanılır.
5. Tabloya hesaplama olarak şu ifadeyi ekleyin:
   ```
   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)))
   ```
   Bu ifade, tablodaki her bir chi2-test fonksiyonunun sonuç değerini, kendisiyle ilişkili yapay boyutun yanına koyma etkisi oluşturur.
6. Hesaplamanın **Sayı biçimlendirmesi** seçeneğini **Sayı** ve **3Anlamlı rakam** olarak ayarlayın.

> *Hesaplamanın ifadesinde bunun yerine şu ifadeyi kullanabilirsiniz:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Sonuç:

Örnek 1 verileri için elde edilen chi2-test fonksiyonları tablosu şu değerleri içerecektir:

Sonuçlar tablosu

p	df	Chi2
0.820	5	2.21

Örnek: Örnek 2

Aşağıdakileri yapın:

1. Örnek 1'de düzenlediğiniz sayfada, **Grafikler**'den bir tablo ekleyin ve **Alanlar**'dan boyutlar olarak Sex, Opinion ve OpCount ekleyin.
2. **Kopyala** ve **Yapıştır** komutlarını kullanarak Örnek 1'den sonuçlar tablosunun kopyasını oluşturun. Hesaplamadaki ifadeyi düzenleyin ve her üç chi2-test fonksiyonundaki bağımsız değişkenleri, Örnek 2 verilerinde kullanılan alanların adlarıyla değiştirin. Örneğin: `Chi2Test_p (Sex,Opinion,OpCount)`.

Sonuç:

Örnek 2 verileri için elde edilen chi2-test fonksiyonları tablosu şu değerleri içerecektir:

Sonuçlar tablosu

p	df	Chi2
0.000309	2	16.2

Örnek: Örnek 3

Aşağıdakileri yapın:

1. Örnek 1 ve Örnek 2 verilerine ilişkin örneklerdeki aynı yöntemle iki tablo daha oluşturun. Boyutlar tablosunda, aşağıdaki alanları boyut olarak kullanın: Gender, Description, Actual ve Expected.
2. Sonuçlar tablosunda, Örnek 3 verilerinde kullanılan alanların adlarını kullanın. Örneğin: `Chi2Test_p (Gender,Description,Actual,Expected)`.

Sonuç:

Örnek 3 verileri için elde edilen chi2-test fonksiyonları tablosu şu değerleri içerecektir:

Sonuçlar tablosu

p	df	Chi2
0.000308	2	16.2

Veri yükleme komut dosyasında chi2-test fonksiyonlarının kullanımına ilişkin örnekler

chi2-test fonksiyonları, ki-kareli istatistiksel analiz ile ilişkili değerleri bulmak için kullanılır. Bu bölümde, Qlik Sense içinde kullanılabilen ki kare dağılımı test fonksiyonlarının veri kod dosyasında nasıl kullanılacağı açıklanmaktadır. Söz dizimi ve bağımsız değişkenler ile ilgili açıklamalar için lütfen ayrı chi2-test kod fonksiyonu konularına bakın.

Bu örnekte, iki öğrenci grubu (I ve II) için not alan (A-F) öğrencilerin sayısını içeren bir tablo kullanılmaktadır.

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Örnek verileri yükleme

Aşağıdakileri yapın:

1. Yeni bir uygulama oluşturun.
2. Veri yükleme düzenleyicisine aşağıdakileri girin:
```
// 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. Verileri yüklemek için seçeneğine tıklayın.

Artık, örnek verileri yüklediniz.

chi2-test fonksiyonu değerlerini yükleme

Şimdi, örnek verileri temel alan chi2-test değerlerini, Grp ölçütüne göre gruplandırılmış olarak yeni bir tabloya yükleyeceğiz.

Aşağıdakileri yapın:

1. Veri yükleme düzenleyicisinde, kodun sonuna aşağıdakileri ekleyin:
```
// 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. Verileri yüklemek için seçeneğine tıklayın.

Böylece, chi2-test değerlerini Chi2_table adında bir tabloya yüklemiş oldunuz.

Sonuçlar

Sonuçta oluşan chi2-test değerlerini **Ön izleme** altındaki veri modeli görüntüleyicisinde görüntüleyebilirsiniz. Şöyle görünmeleri gerekir:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Tipik bir t-test raporu oluşturma

Tipik bir öğrenci t-test raporunda **Group Statistics** ve **Independent Samples Test** sonuçlarını içeren tablolar yer alabilir.

Aşağıdaki bölümlerde, Observation ve Comparison olmak üzere iki bağımsız örnek grubuna uygulanan Qlik Senset-test fonksiyonlarını kullanarak bu tabloları oluşturacağız. Bu örnekler için karşılık gelen tablolar şöyle görünür:

Grup istatistikleri

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

Bağımsız Örnek Testi

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

Örnek verileri yükleme

Aşağıdakileri yapın:

1. Yeni bir sayfayla yeni bir uygulama oluşturun ve bu sayfayı açın.
2. Aşağıdakileri veri yükleme düzenleyicisine girin:

```
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 '|');
```

Bu kod dosyasında, **crosstable** için üç bağımsız değişken gerektiğinden **recno()** dahil edilmiştir. O halde, **recno()** fazladan bir bağımsız değişken sağlar (bu durumda, her bir satır için bir kimlik). Bu olmadan **Comparison** örnek değerleri yüklenemezdi.

3. Verileri yüklemek için seçeneğine tıklayın.

Group Statistics tablosunu oluşturma

Aşağıdakileri yapın:

1. Veri yükleme düzenleyicisinde, seçeneğine tıklayarak uygulama görünümüne gidin ve daha önce oluşturduğunuz sayfaya tıklayın.
   Bu, sayfa görünümünü açar.
2. Sayfayı düzenlemek için **Sayfayı düzenle** seçeneğine tıklayın.
3. **Grafikler**'den bir tablo ekleyin ve **Alanlar**'dan hesaplamalar olarak aşağıdaki ifadeleri ekleyin:

Örnek ifadeler

Etiket	İfade
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Tabloya boyut olarak Type ekleyin.
5. **Sıralama**'ya tıklayın ve Type öğesini sıralama listesinin en üstüne taşıyın.

Sonuç:

Bu örnekler için bir Group Statistics tablosu şöyle görünür:

Grup istatistikleri

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Two Independent Sample Student's T-test tablosunu oluşturma

Aşağıdakileri yapın:

1. Sayfayı düzenlemek için **Sayfayı düzenle** seçeneğine tıklayın.
2. Aşağıdaki ifadeyi tabloya bir boyut olarak ekleyin. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))`
3. **Grafikler**'den hesaplamalar olarak aşağıdaki ifadelerin bulunduğu bir tablo ekleyin:

Örnek ifadeler

Etiket	İfade
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))

Sonuç:

Bağımsız örnek testi

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

z-test fonksiyonlarının kullanımına ilişkin örnekler

z-test fonksiyonları, genellikle 30'dan fazla öğe içeren ve varyansın bilindiği büyük veri örnekleri için z-test istatistiksel analizi ile ilişkili değerleri bulmak amacıyla kullanılır.

Bu bölümde, Qlik Sense içinde kullanılabilen z-test fonksiyonlarının değerlerini bulmak için örnek veriler kullanılarak görselleştirmelerin nasıl oluşturulacağı açıklanmaktadır. Söz dizimi ve bağımsız değişkenler ile ilgili açıklamalar için lütfen, ayrı z-test grafik fonksiyonu konularına bakın.

Örnek verileri yükleme

Burada kullanılan örnek veriler, t-test fonksiyonu örneklerinde kullanılanlar ile aynıdır. Normalde bu örnek veri boyutunun z testi için çok küçük olduğu kabul edilir; ancak Qlik Sense içinde farklı z-test fonksiyonlarının kullanımını gösterme amacı için yeterlidir.

Aşağıdakileri yapın:

1. Yeni bir sayfayla yeni bir uygulama oluşturun ve bu sayfayı açın.

   t-test fonksiyonları için bir uygulama oluşturduysanız o uygulamayı kullanabilir ve bu fonksiyonlar için yeni bir sayfa oluşturabilirsiniz.

2. Veri yükleme düzenleyicisine aşağıdakileri girin:

```
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 '|');
```

Bu kod dosyasında, **crosstable** için üç bağımsız değişken gerektiğinden **recno()** dahil edilmiştir. O halde, **recno()** fazladan bir bağımsız değişken sağlar (bu durumda, her bir satır için bir kimlik). Bu olmadan **Comparison** örnek değerleri yüklenemezdi.

3. Verileri yüklemek için seçeneğine tıklayın.

z-test grafik fonksiyonu görselleştirmelerini oluşturma

Aşağıdakileri yapın:

1. Veri yükleme düzenleyicisinde, seçeneğine tıklayarak uygulama görünümüne gidin ve verileri yüklerken oluşturduğunuz sayfaya tıklayın.
   Sayfa görünümü açılır.
2. Sayfayı düzenlemek için **Sayfayı düzenle** seçeneğine tıklayın.
3. **Grafikler**'den bir tablo ekleyin ve **Alanlar**'dan boyut olarak Type ekleyin.
4. Tabloya hesaplamalar olarak şu ifadeleri ekleyin:

Örnek ifadeler

Etiket	İfade
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)

*Anlamlı değerleri görmek amacıyla hesaplamaların sayı biçimlendirmesini ayarlamak isteyebilirsiniz. Hesaplamaların çoğunda sayı biçimlendirmesini **Auto** yerine **Sayı>Basit** olarak ayarlarsanız tabloyu okumak kolaylaşır. Ancak örneğin, ZTest Sig için **Özel** sayı biçimlendirmesini kullanın ve sonra biçim desenini **# ##** olarak ayarlayın.*

Sonuç:

Örnek veriler için elde edilen z-test fonksiyonları tablosu şu değerleri içerecektir:

Sonuçlar tablosu

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

z-testw grafik fonksiyonu görselleştirmelerini oluşturma

z-testw fonksiyonları, giriş veri serilerinin ağırlıklı iki sütunlu biçimde olduğu durumlarda kullanılmak içindir. İfadelerde, weight bağımsız değişkeni için bir değer gerekir. Buradaki örneklerde hep 2 değeri kullanılmaktadır, ancak her bir gözlem için weight değeri tanımlayacak bir ifade de kullanabilirsiniz.

Örnekler ve sonuçlar:

z-test fonksiyonları için kullanılanın aynı örnek veriler ve sayı biçimlendirmesi kullanıldığında, z-testw fonksiyonları için ortaya çıkan tablo şu değerleri içerecektir:

Sonuçlar tablosu

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

Dize toplama işlevleri

Bu bölümde, dizeyle ilgili toplama işlevleri açıklanmaktadır.

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Veri kod dosyasında dize toplama işlevleri

Concat

Concat(), dize değerlerini birleştirmek için kullanılır. Bu kod fonksiyonu, **group by** cümlesi ile tanımlandığı şekilde, birkaç kayıt üzerinde yinelenen ifadenin tüm değerlerinin toplanmış dize birleşimini döndürür.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue(), bir **group by** cümlesi ile sıralanmış olarak, ifade ile tanımlanan kayıtlardan ilk yüklenmiş değeri döndürür.

Bu fonksiyon yalnızca kod fonksiyonu olarak kullanılabilir.

```
FirstValue  (expression)
```

LastValue

LastValue(), bir **group by** cümlesi ile sıralanmış olarak, ifade ile tanımlanan kayıtlardan son yüklenmiş değeri döndürür.

Bu fonksiyon yalnızca kod fonksiyonu olarak kullanılabilir.

```
LastValue  (expression)
```

MaxString

MaxString() ifade içindeki dize değerlerini bulur ve bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinden alfabetik olarak sıralanan son metin değerini döndürür.

```
MaxString (expression )
```

MinString

MinString(), ifade içindeki dize değerlerini bulur ve bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinde alfabetik olarak sıralanan ilk metin değerini döndürür.

```
MinString (expression )
```

Grafiklerde dize toplama işlevleri

Aşağıdaki grafik fonksiyonları, grafiklerde dizeleri toplamak için kullanılabilir.

Concat

Concat(), dize değerlerini birleştirmek için kullanılır. Fonksiyon, her bir boyut üzerine değerlendirilen ifadenin tüm değerlerinin toplanmış dize birleşimini döndürür.

```
Concat - grafik fonksiyonu({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

MaxString() ifade veya alanda dize değerlerini bulur ve alfabetik sıralamayla son metin değerini döndürür.

```
MaxString - grafik fonksiyonu({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() ifade veya alanda dize değerlerini bulur ve alfabetik sıralamayla ilk metin değerini döndürür.

```
MinString - grafik fonksiyonu({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat(), dize değerlerini birleştirmek için kullanılır. Bu kod fonksiyonu, **group by** cümlesi ile tanımlandığı şekilde, birkaç kayıt üzerinde yinelenen ifadenin tüm değerlerinin toplanmış dize birleşimini döndürür.

Söz Dizimi:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

İşlenecek dizeyi içeren ifade veya alan.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	İşlenecek dizeyi içeren ifade veya alan.
delimiter	Her değer, delimiter içinde bulunan dize ile ayrılabilir.
sort-weight	Birleşimin sırası **sort-weight** boyutunun değerine göre belirlenebilir (varsa) ve en düşük değere karşılık gelen dize birleşimde ilk görünür.
distinct	İfadeden önce **distinct** sözcüğü varsa tüm çoğaltmalar göz ardı edilir.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnekler ve sonuçlar

Örnek	Sonuç	Bir sayfaya eklendikten sonra sonuçlar

`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

Önceki örnekte olduğu gibi **TeamData** tablosunun yüklendiği varsayılırsa:
 	
`LOAD SalesGroup,Concat(distinct Team,'-')`
`as TeamConcat2 Resident TeamData Group By`
`SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat2
		Alpha-Beta-Delta-Gamma
		Epsilon-Eta-Theta-Zeta

Örnek	Sonuç	Bir sayfaya eklendikten sonra sonuçlar
Önceki örnekte olduğu gibi **TeamData** tablosunun yüklendiği varsayılırsa. **sort-weight** için bağımsız değişken eklendiğinden sonuçlar, Amount boyutunun değerine göre sıralanır: `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 - grafik fonksiyonu

Concat(), dize değerlerini birleştirmek için kullanılır. Fonksiyon, her bir boyut üzerine değerlendirilen ifadenin tüm değerlerinin toplanmış dize birleşimini döndürür.

Söz Dizimi:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
string	İşlenecek dizeyi içeren ifade veya alan.
delimiter	Her değer, delimiter içinde bulunan dize ile ayrılabilir.
sort-weight	Birleşimin sırası **sort-weight** boyutunun değerine göre belirlenebilir (varsa) ve en düşük değere karşılık gelen dize birleşimde ilk görünür.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	Fonksiyon bağımsız değişkenlerinden önce **DISTINCT** sözcüğü varsa fonksiyon bağımsız değişkenlerinin değerlendirilmesinden kaynaklanan çoğaltmalar göz ardı edilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

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

İşlev örnekleri

Örnek	Sonuç
`Concat(Team)`	Tablo, SalesGroup ve Amount boyutlarından ve Concat(Team) hesaplaması üzerindeki varyasyonlardan oluşturulmuştur. Toplamlar sonucu yok sayılırsa, sekiz Team değeri için iki SalesGroup değeri geneline yayılmış veriler bulunmasına karşın, Concat(Team) hesaplamasının tabloda birden fazla Team dize değerini birleştiren tek sonucunun Amount 20000 boyutunu içeren satır (BetaGammaGamma sonucunu veren) olduğuna dikkat edin. Bunun nedeni, giriş verilerinde Amount 20000 için üç değer bulunmasıdır. Hesaplama boyutlar geneline yayıldığında tüm diğer sonuçlar birleştirilmeden kalır; çünkü her SalesGroup ve Amount kombinasyonu için yalnızca bir Team değeri vardır.
`Concat (DISTINCT Team,', ')`	Beta, Gamma. Çünkü DISTINCT niteleyicisi, çoğaltma Gamma sonucunun göz ardı edilmesi anlamına gelir. Ayrıca, sınırlayıcı bağımsız değişken, virgül ve bunu izleyen boşluk olarak tanımlanır.
`Concat (TOTAL <SalesGroup> Team)`	TOTAL niteleyicisi kullanılırsa, tüm Team değerleri için tüm dize değerleri birleştirilir. Alan seçimi <SalesGroup> belirtildiğinde, sonuçları SalesGroup boyutunun iki değeri halinde böler. SalesGroupEast için, sonuçlar AlphaBetaDeltaGammaGamma olur. SalesGroupWest için, sonuçlar EpsilonEtaThetaZeta olur.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	**sort-weight**: Amount için bağımsız değişken eklenerek sonuçlar Amount boyutunun değerine göre sıralanır. Sonuçlar DeltaBetaGammaGammaAlpha ve EtaEpsilonZEtaTheta olur.

Örnekte kullanılan veriler:

```
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(), bir **group by** cümlesi ile sıralanmış olarak, ifade ile tanımlanan kayıtlardan ilk yüklenmiş değeri döndürür.

Bu fonksiyon yalnızca kod fonksiyonu olarak kullanılabilir.

Söz Dizimi:

```
FirstValue ( expr)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değeri bulunmuyorsa NULL döndürülür.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç	Bir sayfadaki sonuçlar
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(), bir **group by** cümlesi ile sıralanmış olarak, ifade ile tanımlanan kayıtlardan son yüklenmiş değeri döndürür.

Bu fonksiyon yalnızca kod fonksiyonu olarak kullanılabilir.

Söz Dizimi:

```
LastValue ( expr )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değeri bulunmuyorsa NULL döndürülür.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamamızdaki bir sayfaya ekleyin.

Aşağıdaki sonuç sütunuyla aynı görünümü elde etmek için özellikler panelinde, Sıralama altında, Otomatik'ten Özel'e geçin ve sayısal ve alfabetik sıralamanın seçimini kaldırın.

Örnek	Sonuç	Özel sıralama ile sonuç
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() ifade içindeki dize değerlerini bulur ve bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinden alfabetik olarak sıralanan son metin değerini döndürür.

Söz Dizimi:

```
MaxString ( expr )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değeri bulunmuyorsa NULL döndürülür.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Örnek	Sonuç	
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
Önceki örnekte olduğu gibi **TeamData** tablosunun yüklendiği ve veri yükleme kod dosyanızda SET deyiminin bulunduğu varsayılırsa: 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 - grafik fonksiyonu

MaxString() ifade veya alanda dize değerlerini bulur ve alfabetik sıralamayla son metin değerini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.

Bağımsız Değişken	Açıklama
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Sınırlamalar:

İfadede dize temsiline sahip hiçbir değer yoksa, NULL döndürülür.

Örnekler ve sonuçlar:

Sonuçlar tablosu

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

İşlev örnekleri

Örnek	Sonuç
`MaxString (Team)`	Amount boyutu için üç 20000 değeri bulunmaktadır: ikisi Gamma (farklı tarihlerde) ve biri Beta. Dolayısıyla, MaxString (Team) hesaplamasının sonucu Gamma olur; çünkü sıralanan dizelerdeki en yüksek değer budur.
`MaxString (Date)`	2013/11/01, Amount boyutuyla ilişkili olarak üçü arasında en büyük Date değeridir. Burada, kodunuzda SET deyiminin olduğu varsayılmaktadır `SET DateFormat='YYYY-MM-DD';`

Örnekte kullanılan veriler:

```
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(), ifade içindeki dize değerlerini bulur ve bir **group by** cümlesi ile tanımlandığı şekilde bir dizi kayıt üzerinde alfabetik olarak sıralanan ilk metin değerini döndürür.

Söz Dizimi:

```
MinString ( expr )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Sınırlamalar:

Metin değeri bulunmuyorsa NULL döndürülür.

Örnekler ve sonuçlar:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Sonuç verileri

Örnek	Sonuç	
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

Örnek	Sonuç	
Önceki örnekte olduğu gibi **TeamData** tablosunun yüklendiği ve veri yükleme kod dosyanızda SET deyiminin bulunduğu varsayılırsa: `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 - grafik fonksiyonu

MinString() ifade veya alanda dize değerlerini bulur ve alfabetik sıralamayla ilk metin değerini döndürür.

Söz Dizimi:

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

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
TOTAL	**TOTAL** sözcüğü, fonksiyon bağımsız değişkenlerinden önce gelirse, hesaplama yalnızca geçerli boyutsal değere ait olanlar için değil, geçerli seçimlerde verilen tüm olası değerler üzerinden yapılır; yani grafik boyutlarını göz ardı eder. **TOTAL** niteleyicisinden sonra grafik boyut değişkenlerinin bir alt kümesi olarak bir veya daha çok alan adının geldiği **TOTAL [<fld {.fld}>]** niteleyicisini kullanarak toplam olası değerlerin bir alt kümesini oluşturursunuz.

Örnekler ve sonuçlar:

Örnek veriler

SalesGroup	Amount	MinString(Team)	MinString(Date)
East	14000	Delta	2013/08/01
East	20000	Beta	2013/05/01
East	25000	Alpha	2013/07/01

SalesGroup	Amount	MinString(Team)	MinString(Date)
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

İşlev örnekleri

Örnekler	Sonuçlar
`MinString (Team)`	Amount boyutu için üç 20000 değeri bulunmaktadır: ikisi Gamma (farklı tarihlerde) ve biri Beta. Dolayısıyla, MinString (Team) hesaplamasının sonucu Beta olur; çünkü sıralanan dizelerdeki ilk değer budur.
`MinString (Date)`	2013/11/01, Amount boyutuyla ilişkili olarak üçü arasında en erken Date değeridir. Burada, kodunuzda SET deyiminin olduğu varsayılmaktadır `SET DateFormat='YYYY-MM-DD';'`

Örnekte kullanılan veriler:

```
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 '|');
```

Yapay boyut fonksiyonları

Yapay boyut, uygulamada, doğrudan veri modelindeki alanlardan değil de, yapay boyut fonksiyonlarından üretilen değerlerden oluşturulur. Yapay boyut fonksiyonu ile üretilen değerler bir grafikte hesaplanan boyut olarak kullanıldığında, bu bir yapay boyut oluşturur. Yapay boyutlar, örneğin, verilerinizden gelen değerlere sahip boyutları (yani, dinamik boyutları) içeren grafikler oluşturmanıza izin verir.

Yapay boyutlar seçimlerden etkilenmez.

Aşağıdaki yapay boyut fonksiyonları grafiklerde kullanılabilir.

ValueList

ValueList(), hesaplanan boyutta kullanıldığında yapay bir boyut oluşturacak olan listelenmiş değerler kümesini döndürür.

```
ValueList - grafik fonksiyonu (v1 {, Expression})
```

ValueLoop

ValueLoop(), hesaplanan boyutta kullanıldığında yapay bir boyut oluşturacak olan yinelenen değerler kümesini döndürür.

```
ValueLoop - grafik fonksiyonu(from [, to [, step ]])
```

ValueList - grafik fonksiyonu

ValueList(), hesaplanan boyutta kullanıldığında yapay bir boyut oluşturacak olan listelenmiş değerler kümesini döndürür.

> ***ValueList*** *fonksiyonuyla oluşturulmuş yapay boyutlu grafiklerde, grafik ifadesindeki aynı parametrelerle* ***ValueList*** *fonksiyonunu yeniden belirterek belirli bir ifade hücresine karşılık gelen boyut değerine referansta bulunabilir. Bu fonksiyon, tabii ki, düzen içinde herhangi bir yerde kullanılabilir, ancak, yapay boyutlar için kullanıldığı zamanlar dışında, yalnızca toplama işlevi içinde anlamlı olur.*

> *Yapay boyutlar seçimlerden etkilenmez.*

Söz Dizimi:

```
ValueList(v1 {,...})
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
v1	Statik değer (genellikle bir dizedir; ancak sayı da olabilir).
{,...}	İsteğe bağlı statik değerler listesi.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnek	Sonuç
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Tabloda bir boyut oluşturmak için kullanıldığında, bu örneğin, üç dize değerinin tablodaki satır etiketleri olmasıyla sonuçlanır. Daha sonra bir ifade içinde bunlara referansta bulunulabilir.

Örnek	Sonuç
`=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) ))`	Bu ifade, değerleri oluşturulan boyuttan alır ve üç toplama işlevi için giriş olarak, iç içe bir IF deyiminde bunlara referansta bulunur: **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

Örneklerde kullanılan veriler:

```
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 - grafik fonksiyonu

ValueLoop(), hesaplanan boyutta kullanıldığında yapay bir boyut oluşturacak olan yinelenen değerler kümesini döndürür.
Oluşturulmuş değerler, adım artırımlı ara değerler de dahil olmak üzere, **from** değeriyle başlayıp **to** değeriyle biter.

> ***ValueLoop*** *fonksiyonuyla oluşturulmuş yapay boyutlu grafiklerde, grafik ifadesindeki aynı parametrelerle* ***ValueLoop*** *fonksiyonunu yeniden belirterek belirli bir ifade hücresine karşılık gelen boyut değerine referansta bulunabilir. Bu fonksiyon, tabii ki, düzen içinde herhangi bir yerde kullanılabilir, ancak, yapay boyutlar için kullanıldığı zamanlar dışında, yalnızca toplama işlevi içinde anlamlı olur.*

Yapay boyutlar seçimlerden etkilenmez.

Söz Dizimi:

```
ValueLoop(from [, to [, step ]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
from	Oluşturulacak değerler kümesinde başlangıç değeri.
to	Oluşturulacak değerler kümesinde bitiş değeri.
step	Değerler arasında artış boyutu.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnek	Sonuç
`ValueLoop (1, 10)`	Bu örneğin, tabloda, numaralandırılmış etiketleme gibi amaçlarla kullanılabilecek bir boyut oluşturur. Buradaki örnek, 1 ila 10 olarak numaralandırılmış değerleri verir. Daha sonra bir ifade içinde bu değerlere referansta bulunabilir.
`ValueLoop (2, 10,2)`	Bu örnek, 2, 4, 6, 8 ve 10 olarak numaralandırılmış değerleri verir; çünkü step bağımsız değişkeninin değeri 2'dir.

İç içe geçmeli toplamalar

Bir toplamayı başka bir toplamanın sonucuna uygulamanız gereken durumlarla karşılaşabilirsiniz. Bu uygulama iç içe geçmeli toplamalar olarak adlandırılır.

Çoğu grafik ifadesinde toplamaları iç içe geçiremezsiniz. Ancak iç toplama fonksiyonunda **TOTAL** niteleyicisini kullanırsanız toplamaları iç içe geçirebilirsiniz.

En fazla 100 düzeyde iç içe geçmeye izin verilir.

TOTAL niteleyicili iç içe geçmeli toplamalar

Örnek:

Sales alanının toplamını hesaplamak, ancak yalnızca **OrderDate** alanı geçen yıla eşit olan işlemleri dahil etmek istiyorsunuz. Geçen yıl, **`Max(TOTAL Year(OrderDate))`** toplama işleviyle elde edilebilir.

Aşağıdaki toplama işlevi istenen sonucu döndürecektir:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense, bu tür bir iç içe geçirme için **TOTAL** niteleyicisinin eklenmesini gerektirir. İstenen karşılaştırma için gereklidir. Bu tür iç içe geçme ihtiyacı oldukça yaygındır ve iyi bir uygulamadır.

Ayrıca bkz.

Aggr - grafik fonksiyonu (page 410)

5.3 Aggr - grafik fonksiyonu

Aggr(), belirtilen boyut veya boyutlar üzerinde hesaplanan ifade için bir değer dizisi döndürür. Örneğin, her bölge için müşteri başına maksimum satış değeri.

Aggr işlevi, ilk parametresinin (iç toplama) her boyutsal değer için bir kez hesaplandığı iç içe geçmiş toplamalar için kullanılır. Boyutlar ikinci parametrede (ve sonraki parametrelerde) belirtilir.

Ayrıca **Aggr** işlevinin sonuç dizisi, içinde bulunduğu toplamaya girdi olarak kullanılarak **Aggr** işlevi bir dış toplama işlevinin içine alınmalıdır.

Söz Dizimi:

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

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Toplama işlevinden oluşan bir ifade. Toplama işlevi varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır.
StructuredParameter	StructuredParameter, bir boyuttan ve isteğe bağlı olarak şu biçimdeki bir sıralama ölçütünden oluşur: `(Dimension(Sort-type, Ordering))` Boyut tek bir alandır ve ifade olamaz. Boyut, Aggr ifadesinin hesaplandığı değer dizisini belirlemek için kullanılır. Sıralama ölçütleri dahil edilirse Aggr fonksiyonu tarafından oluşturulan ve boyut için hesaplanan değer dizisi sıralanır. Bu, sıralama düzeni Aggr fonksiyonunun bulunduğu ifadenin sonucunu etkilediğinde önemli olur. Sıralama ölçütlerinin nasıl kullanılacağıyla ilgili ayrıntılar için bkz. Yapılandırılmış parametrede boyuta sıralama ölçütleri ekleme.

Bağımsız Değişken	Açıklama
SetExpression	Toplama işlevi, varsayılan olarak, seçim tarafından tanımlanmış olası kayıtlar kümesi üzerinden toplanır. Bir set analizi ifadesi ile alternatif bir kayıt kümesi tanımlanabilir.
DISTINCT	İfade bağımsız değişkeninden önce **distinct** niteleyicisi geliyorsa veya hiçbir niteleyici kullanılmamışsa, boyut değerlerinin her bir tekil kombinasyonu yalnızca bir döndürülen değer üretir. Toplamalar normalde bu yolla yapılır; boyut değerlerinin her bir tekil kombinasyonu, grafikteki bir çizgiyi oluşturur.
NODISTINCT	İfade bağımsız değişkeninden önce **nodistinct** niteleyicisi geliyorsa boyut değerlerinin her bir birleşimi, temel veri yapısına bağlı olarak, birden fazla döndürülen değer üretir. Yalnızca tek bir boyut varsa **aggr** fonksiyonu, kaynak verilerdeki satır sayısı ile aynı sayıda öğe içeren bir dizi döndürür.

Sum, **Min** ve **Avg** gibi temel toplama işlevleri tek bir sayısal değer döndürürken, Aggr() fonksiyonu, başka bir toplamanın gerçekleşebileceği geçici, aşamalandırılmış bir sonuç kümesi (sanal tablo) oluşturulmasıyla karşılaştırılabilir. Örneğin, ortalama satış değerini hesaplamak için, bir **Aggr()** deyimi içinde müşteri bazında satışların toplamının alınması ve sonra da toplamı alınan bu sonuçların ortalamasının hesaplanması: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Birden fazla düzey halinde iç içe geçmiş grafik toplamaları oluşturmak isterseniz hesaplanan boyutlarda Aggr() fonksiyonunu kullanın.

Sınırlamalar:

Aggr() fonksiyonundaki her boyut tek bir alan olmalıdır ve bir ifade (hesaplanan boyut) olamaz.

Yapılandırılmış parametrede boyuta sıralama ölçütleri ekleme

Temel biçiminde, Aggr fonksiyon söz dizimindeki StructuredParameter bağımsız değişkeni tek bir boyuttur. İfade: Aggr(Sum(Sales, Month)), her bir ay için toplam satış değerini bulur. Ancak, başka bir toplama işlevine dahil edildiğinde, sıralama ölçütleri kullanılmazsa beklenmedik sonuçlar ortaya çıkabilir. Bunun nedeni, bazı boyutların sayısal veya alfabetik olarak sıralanması, vb. olabilir.

Aggr fonksiyonundaki StructuredParameter bağımsız değişkeninde, ifadenizdeki boyutta sıralama ölçütlerini belirtebilirsiniz. Bu şekilde, Aggr fonksiyonu tarafından oluşturulan sanal tabloda bir sıralama düzeni uygularsınız.

StructuredParameter bağımsız değişkeni aşağıdaki söz dizimine sahiptir:

```
(FieldName, (Sort-type, Ordering))
```

Yapılandırılmış parametreler iç içe geçebilir:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Sıralama türü şunlar olabilir: NUMERIC, TEXT, FREQUENCY veya LOAD_ORDER.

Her Sıralama türüyle ilişkilendirilen Düzenleme türleri şöyledir:

İzin verilen düzenleme türleri

Sıralama türü	İzin verilen Düzenleme türleri
NUMERIC	ASCENDING, DESCENDING veya REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE veya Z2A
FREQUENCY	DESCENDING, REVERSE veya ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING veya REVERSE

REVERSE ve DESCENDING düzenleme türleri eşdeğerdir.

TEXT sıralama türü için ASCENDING ve A2Z düzenleme türleri ile DESCENDING, REVERSE ve Z2A eşdeğerdir.

LOAD_ORDER sıralama türü için ASCENDING ve ORIGINAL düzenleme türleri eşdeğerdir.

Örnekler: Toplama kullanan grafik ifadeleri

Örnekler - grafik ifadeleri

Grafik ifadesi örneği 1

Komut dosyası

Aşağıdaki grafik ifadesi örneğini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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 '|');
```

Grafik ifadesi

Qlik Sense sayfasında bir KPI görselleştirmesi oluşturun. Şu ifadeyi bir hesaplama olarak KPI'ya ekleyin:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Sonuç

376.7

Açıklama

`Aggr(Sum(UnitSales*UnitPrice), Customer)` ifadesi, **Customer** bazında toplam satış değerini bulur ve şu değerlerden oluşan bir dizi döndürür: üç **Customer** değeri için 295, 715, ve 120.

Değerleri içeren özel bir tablo veya sütun oluşturmak zorunda kalmadan etkili bir şekilde değerlerin geçici listesini oluşturduk.

Bu değerler **Avg()** fonksiyonu için giriş olarak kullanılır ve satışların ortalama değeri olarak 376.7 bulunur.

Grafik ifadesi örneği 2

Komut dosyası

Aşağıdaki grafik ifadesi örneğini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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 '|');
```

Grafik ifadesi

Qlik Sense sayfasında **Customer**, **Product**, **UnitPrice** ve **UnitSales** alanlarını boyut olarak kullanarak bir tablo görselleştirmesi oluşturun. Tabloya hesaplama olarak şu ifadeyi ekleyin:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Sonuç

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

Açıklama

Değer dizisi: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 ve 19. **nodistinct** niteleyicisi, dizinin kaynak verilerindeki her satır için bir öğe içerdiği anlamına gelir: her biri, her **Customer** ve **Product** için maksimum **UnitPrice** değeridir.

Grafik ifadesi örneği 3

Komut dosyası

Aşağıdaki grafik ifadesi örneğini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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;
```

Grafik ifadeleri

Qlik Sense sayfasında **Year** ve **Month** alanlarını boyut olarak kullanarak bir tablo görselleştirmesi oluşturun. Tabloya hesaplama olarak şu ifadeleri ekleyin:

- `Sum(Sales)`
- Tabloda Structured Aggr() olarak etiketlenmiş `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`.

Sonuç

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

Açıklama

Bu örnek, her yıl için on iki aylık bir dönemden toplanan değerleri zaman sırasına göre artan düzende göstermektedir. Yapısal parametrelerin (Numeric, Ascending) **Aggr()** ifadesine dahil edilmesinin nedeni budur. Yapısal parametre olarak iki spesifik boyut gerekir: **Year** ve **Month**, sıralanmış olarak (1) **Year** (sayısal) ve (2) **Month** (sayısal). Bu iki boyut, tabloda veya grafik görselleştirmesinde kullanılmalıdır. Bu, **Aggr()** işlevinin boyut listesinin görselleştirmede kullanılan nesnenin boyutlarına uygun olması için gereklidir.

Bir tabloda veya ayrı çizgi grafiklerde bu hesaplamalar arasındaki farkı karşılaştırabilirsiniz:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Sadece ikinci ifadenin toplama değerlerinin istendiği gibi biriktirilmesini sağlayacağı açıkça anlaşılmalıdır.

Ayrıca bkz.

Temel toplama işlevleri (page 210)

5.4 Renk fonksiyonları

Bu fonksiyonlar, hem grafik nesnelerinin renk özelliklerinin ayarlanması ve değerlendirilmesi ile ilişkili ifadelerde hem de veri kod dosyalarında kullanılabilir.

> *Qlik Sense, geriye dönük uyumluluk gerekçesiyle **Color()**, **qliktechblue** ve **qliktechgray** renk fonksiyonlarını destekler, ancak bunların kullanılması önerilmez.*

ARGB

ARGB(), ifadelerde bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için kullanılır. Burada renk bir kırmızı bileşen **r**, bir yeşil bileşen **g** ve bir mavi bileşen **b** ile tanımlanır (alfa faktörü (opaklık) **alpha** kullanımıyla).

```
ARGB (alpha, r, g, b)
```

HSL

HSL(), bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için ifadelerde kullanılır; burada renk, 0 ile 1 arasındaki **hue**, **saturation** ve **luminosity** değerleriyle tanımlanır.

```
HSL (hue, saturation, luminosity)
```

RGB

RGB(), kırmızı bileşeni r, yeşil bileşeni g ve mavi bileşeni b olmak üzere üç bileşenle tanımlanan rengin koduna karşılık gelen bir tam sayı döndürür. Bu bileşenlerin 0 ile 255 arasında tam sayı değerleri olmalıdır. İşlev, bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için ifadelerde kullanılabilir.

```
RGB (r, g, b)
```

Colormix1

Colormix1() ifadelerde, 0 ile 1 arasında bir değeri temel alan iki renkli gradyandan bir ARGB renk temsili döndürmek için kullanılır.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value, 0 ile 1 arasında gerçek bir sayıdır.

- Value = 0 ise ColorZero döndürülür.
- Value = 1 ise ColorOne döndürülür.
- 0 < Value < 1 ise uygun ara gölgelendirme döndürülür.

ColorZero, aralığın düşük ucuyla ilişkilendirilecek renk için geçerli bir RGB renk temsilidir.

ColorOne, aralığın yüksek ucuyla ilişkilendirilecek renk için geçerli bir RGB renk temsilidir.

Örnek:

```
Colormix1(0.5, red(), blue())
```

şunu döndürür:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Colormix2() fonksiyonu ifadelerde, -1 ile 1 arasında bir değeri temel alan ve merkezi konum (0) için bir ara renk belirtme olasılığı bulunan iki renkli gradyandan bir ARGB renk temsili döndürmek için kullanılır.

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value, -1 ile 1 arasında gerçek bir sayıdır.

- Value = -1 ise ilk renk döndürülür.
- Value = 1 ise ikinci renk döndürülür.
- -1 < Value< 1 ise uygun renk karışımı döndürülür.

ColorMinusOne, aralığın düşük ucuyla ilişkilendirilecek renk için geçerli bir RGB renk temsilidir.

ColorOne, aralığın yüksek ucuyla ilişkilendirilecek renk için geçerli bir RGB renk temsilidir.

ColorZero, aralığın merkeziyle ilişkilendirilecek renge yönelik isteğe bağlı ve geçerli bir RGB renk temsilidir.

SysColor

SysColor(), Windows sistem rengi nr için ARGB renk temsilini döndürür; burada nr, Windows API fonksiyonuna (**GetSysColor(nr)**) yönelik parametreye karşılık gelir.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue(), HSV renk modelinin ton bileşenini değiştiren renk eşlemesinden rengin bir ARGB değerini döndürür. Renk eşlemesi kırmızı ile başlar, sarı, yeşil, camgöbeği, mavi, eflatundan geçer ve kırmızıya döner. x 0 ile 1 arasında bir değer olarak belirtilmelidir.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet(), mavi ile başlayan, camgöbeği, sarı ve turuncudan geçen ve kırmızıya dönen bir renk eşlemesinden bir rengin ARGB değerini döndürür. x 0 ile 1 arasında bir değer olarak belirtilmelidir.

```
ColorMapJet (x)
```

Önceden tanımlanmış renk fonksiyonları

Aşağıdaki fonksiyonlar, önceden tanımlanmış renkler için ifadelerde kullanılabilir. Her bir fonksiyon bir RGB renk temsilini döndürür.

İsteğe bağlı olarak, alfa faktörü için bir parametre verilebilir ve bu durumda bir ARGB renk temsili döndürülür. 0 değerli alfa faktörü tam şeffaflığa karşılık gelirken, 255 değerli alfa faktörü tam opaklığa karşılık gelir. Alfa için bir değer girilmezse 255 olduğu varsayılır.

Önceden tanımlanmış renk fonksiyonları

Renk fonksiyonu	RGB değeri
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)

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB(), ifadelerde bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için kullanılır. Burada renk bir kırmızı bileşen **r**, bir yeşil bileşen **g** ve bir mavi bileşen **b** ile tanımlanır (alfa faktörü (opaklık) **alpha** kullanımıyla).

Söz Dizimi:

```
ARGB(alpha, r, g, b)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
alpha	0-255 aralığında şeffaflık değeri. 0 tam şeffaflığa karşılık gelirken, 255 tam opaklığa karşılık gelir.
r, g, b	Kırmızı, yeşil ve mavi bileşen değerleri. Bir renk bileşeninin 0 olması hiç katkı olmamasına ve 255 olması da tam katkıya karşılık gelir.

Tüm bağımsız değişkenler 0 ila 255 aralığında tamsayılara çözülen ifadeler olmalıdır.

Sayısal bileşen yorumlanıyorsa ve onaltılık gösterimde biçimlendiriliyorsa, renk bileşenlerinin değerlerini görmek daha kolay olur. Örneğin, açık yeşilin numarası 4 278 255 360'tır ve bu değer onaltılık gösterimde FF00FF00 olur. İlk iki konum olan 'FF' (255), **alpha** kanalını belirtir. Sonraki iki konum olan '00', **kırmızı** miktarını, sonraki iki konum olan 'FF', **yeşil** miktarını ve son iki konum olan '00', **mavi** miktarını gösterir.

RGB

RGB(), kırmızı bileşeni r, yeşil bileşeni g ve mavi bileşeni b olmak üzere üç bileşenle tanımlanan rengin koduna karşılık gelen bir tam sayı döndürür. Bu bileşenlerin 0 ile 255 arasında tam sayı değerleri olmalıdır. İşlev, bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için ifadelerde kullanılabilir.

Söz Dizimi:

```
RGB (r, g, b)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
r, g, b	Kırmızı, yeşil ve mavi bileşen değerleri. Bir renk bileşeninin 0 olması hiç katkı olmamasına ve 255 olması da tam katkıya karşılık gelir.

Tüm bağımsız değişkenler 0 ila 255 aralığında tamsayılara çözülen ifadeler olmalıdır.

Sayısal bileşen yorumlanıyorsa ve onaltılık gösterimde biçimlendiriliyorsa, renk bileşenlerinin değerlerini görmek daha kolay olur. Örneğin, açık yeşilin numarası 4 278 255 360'tır ve bu değer onaltılık gösterimde FF00FF00 olur. İlk iki konum olan 'FF' (255), **alpha** kanalını belirtir. **RGB** ve **HSL** fonksiyonlarında bu her zaman 'FF' (opak) olur. Sonraki iki konum olan '00', **kırmızı** miktarını, sonraki iki konum olan 'FF', **yeşil** miktarını ve son iki konum olan '00', **mavi** miktarını gösterir.

Örnek: Grafik ifadesi
Bu örnek bir grafiğe özel bir renk uygular:

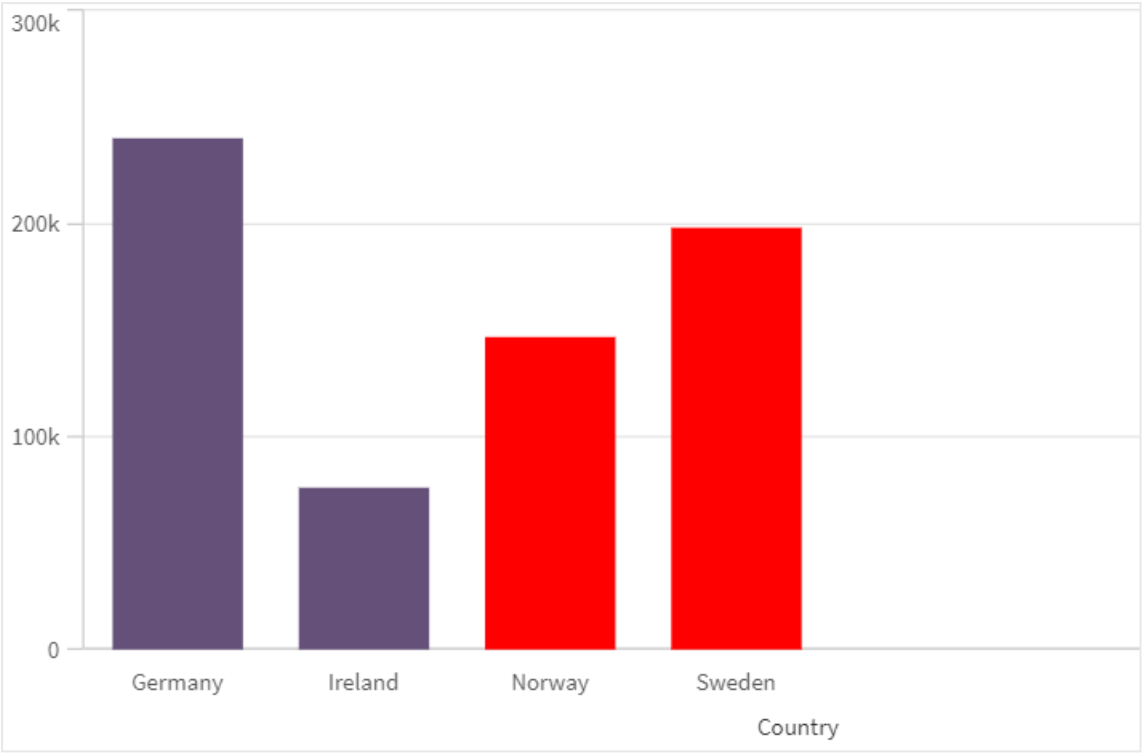
Bu örnekte kullanılan veriler:

```
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 ',');
```

Renkler ve gösterge özellikler paneline aşağıdaki ifadeyi girin:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Sonuç:



Örnek: Yükleme kodu

Aşağıdaki örnek, onaltılık biçimdeki değerleri eşdeğer RGB değerleriyle görüntüler:

```
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];
```

Sonuç:

Metin	Renk
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL(), bir grafik nesnesinin renk özelliklerini ayarlamak veya değerlendirmek için ifadelerde kullanılır; burada renk, 0 ile 1 arasındaki **hue**, **saturation** ve **luminosity** değerleriyle tanımlanır.

Söz Dizimi:

```
HSL (hue, saturation, luminosity)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
hue, saturation, luminosity	0 ile 1 arasında değişen hue, saturation ve luminosity bileşen değerleri.

Tüm bağımsız değişkenler 0 ila 1 aralığında tamsayılara çözülen ifadeler olmalıdır.

Sayısal bileşen yorumlanıyorsa ve onaltılık gösterimde biçimlendiriliyorsa, renk bileşenlerinin RGB değerlerini görmek daha kolay olur. Örneğin, açık yeşilin numarası 4 278 255 360'tır ve bu değer onaltılık gösterimde FF00FF00 ve RGB (0,255,0) olur. Bu da HSL (80/240, 240/240, 120/240) (HSL değeri (0.33, 1, 0.5)) ile eşdeğerdir.

5.5 Koşullu fonksiyonlar

Tüm koşullu fonksiyonlar bir koşulu değerlendirir ve ardından, koşul değerine bağlı olarak farklı yanıtlar döndürür. Fonksiyonlar veri kod dosyasında ve grafik ifadelerinde kullanılabilir.

Koşullu fonksiyonlara genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

alt

alt fonksiyonu, geçerli bir sayı temsiline sahip olan parametrelerin ilkini döndürür. Böyle bir eşleşme bulunamazsa, son parametre döndürülür. Herhangi bir sayıda parametre kullanılabilir.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

class fonksiyonu ilk parametreyi bir sınıf aralığına atar. Sonuçta, metin değeri olarak a<=x<b'nin bulunduğu ikili bir değer elde edilir. Burada a ve b, bölmenin alt ve üst sınırları ve sayısal değer olarak düşük sınırdır.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

coalesce fonksiyonu, geçerli bir non-NULL temsiline sahip olan parametrelerin ilkini döndürür. Herhangi bir sayıda parametre kullanılabilir.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

if fonksiyonu, fonksiyon ile sağlanan koşulun True ya da False olarak değerlendirilmesine göre bir değer döndürür.

```
if (condition , then , else)
```

match

match fonksiyonu, ilk parametreyi ondan sonra gelen tüm parametrelerle karşılaştırır ve eşleşen ifadelerin sayısal konumunu döndürür. Karşılaştırma büyük/küçük harf duyarlıdır.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

mixmatch fonksiyonu, ilk parametreyi ondan sonra gelen tüm parametrelerle karşılaştırır ve eşleşen ifadelerin sayısal konumunu döndürür. Karşılaştırma büyük/küçük harf duyarlı değildir.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

Pick fonksiyonu listedeki *n*. ifadeyi döndürür.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

wildmatch fonksiyonu ilk parametreyi sonraki tüm parametrelerle karşılaştırır ve eşleşen ifadenin sayısını döndürür. Karşılaştırma dizelerinde joker karakterlerin (* ve ?) kullanılmasına izin verir. *, herhangi bir karakter sırasını eşleştirir. ?, herhangi bir tek karakterle eşleşir. Karşılaştırma büyük/küçük harf duyarlı değildir.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

alt fonksiyonu, geçerli bir sayı temsiline sahip olan parametrelerin ilkini döndürür. Böyle bir eşleşme bulunamazsa, son parametre döndürülür. Herhangi bir sayıda parametre kullanılabilir.

Söz Dizimi:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr1	Geçerli bir sayı temsili denetimi için ilk ifade.
expr2	Geçerli bir sayı temsili denetimi için ikinci ifade.
expr3	Geçerli bir sayı temsili denetimi için üçüncü ifade.
else	Önceki parametrelerin hiçbirinde geçerli bir sayı temsili olmaması durumunda dönen değer.

alt fonksiyonu çoğu zaman sayı veya tarih yorumlama fonksiyonları ile birlikte kullanılır. Bu sayede Qlik Sense, önceliği belirlenmiş bir sırada farklı tarih biçimlerini test edebilir. Ayrıca, sayısal ifadelerde NULL değerleri işlemek için de kullanılabilir.

Örnekler:

Örnekler

Örnek	Sonuç

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	Bu ifade, tarih alanının belirtilen üç tarih biçiminden herhangi birine göre bir tarih içerip içermediğini test eder. İçerdiği takdirde, ilk dizeyi ve tarihin geçerli bir sayı temsilini içeren ikili bir değer döndürür. Bir eşleşme bulunamazsa, 'No valid date' metni döndürülür (herhangi bir geçerli sayı temsili olmadan).

`alt(Sales,0) + alt(Margin,0)`	Bu ifade Sales ve Margin alanlarını ekler ve eksik (NULL) değerlerin yerine 0 koyar.

class

class fonksiyonu ilk parametreyi bir sınıf aralığına atar. Sonuçta, metin değeri olarak a<=x<b'nin bulunduğu ikili bir değer elde edilir. Burada a ve b, bölmenin alt ve üst sınırları ve sayısal değer olarak düşük sınırdır.

Söz Dizimi:

```
class(expression, interval [ , label [ , offset ]])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
interval	Bölme genişliğini belirten bir sayı.
label	Sonuç metninde 'x' değerinin yerini alabilen rastgele seçilmiş bir dize.
offset	Sınıflandırmanın varsayılan başlangıç noktasından kaydırma olarak kullanılabilecek bir sayı. Varsayılan başlangıç noktası normalde 0'dır.

Örnekler:

Örnekler

Örnek	Sonuç
`var = 23` ile `class( var,10 )`	şunu döndürür: '20<=x<30'
`var = 23` ile `class( var,5,'value' )`	şunu döndürür: '20<= value <25'
`var = 23` ile `class( var,10,'x',5 )`	şunu döndürür: '15<=x<25'

Örnek - class kullanan yükleme kodu

Örnek: yükleme kodu

Komut dosyası

Bu örnekte, insanların adını ve yaşını içeren bir tablo yüklüyoruz. Tek tek herkesi on yıl aralıkla yaş gruplarına sınıflandıran bir alan eklemek istiyoruz. Özgün kaynak tablo aşağıdaki gibi görünür.

Sonuçlar

Name	Age
John	25
Karen	42
Yoshi	53

Yaş grubu sınıflandırma alanını eklemek için **class** fonksiyonunu kullanarak bir öncelikli yükleme deyimi ekleyebilirsiniz.

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Sonuçları görmek için Qlik Sense uygulamasında tabloyu oluşturun.

```
 LOAD *, class(Age, 10, 'age') As Agegroup;  LOAD * INLINE [ Age, Name 25, John 42, Karen 53, Yoshi];
```

Sonuçlar

Sonuçlar

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

coalesce fonksiyonu, geçerli bir non-NULL temsiline sahip olan parametrelerin ilkini döndürür. Herhangi bir sayıda parametre kullanılabilir.

Söz Dizimi:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr1	NULL olmayan geçerli bir gösterimi kontrol edecek ilk ifade.
expr2	NULL olmayan geçerli bir gösterimi kontrol etmek için ikinci ifade.
expr3	NULL olmayan geçerli bir gösterimi kontrol etmek için üçüncü ifade.

Örnekler:

Örnekler

Örnek	Sonuç
	Bu ifade, bir alanın tüm NULL değerlerini "Yok" olarak değiştirir.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Bu ifade, bazı alanların ürün için değerlere sahip olmadığı durumlarda üç farklı ürün açıklama alanı arasından seçim yapar. Null olmayan bir değere sahip alanlardan ilki verilen sırayla döndürülür. Alanlardan hiçbiri değer içermiyorsa, sonuç "açıklama yok" olacaktır.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Bu ifade, *FileName* alanında olabilecek kapsayıcı tırnak işaretlerini kesecektir. Belirtilen *FileName* tırnak içine alınmışsa bunlar kaldırılır ve *FileName* ayraç içine alınmış, tırnak işaretleri kaldırılmış biçimde döndürülür. *TextBetween* fonksiyonu sınırlayıcıları bulamazsa, **Coalesce** tarafından reddedilen null değerini döndürür ve bunun yerine ham *FileName* döndürür.

if

if fonksiyonu, fonksiyon ile sağlanan koşulun True ya da False olarak değerlendirilmesine göre bir değer döndürür.

Söz Dizimi:

```
if(condition , then [, else])
```

if fonksiyonunun *condition*, *then* ve *else* şeklinde üç parametresi vardır ve bunların tümü birer ifadedir. Diğer iki parametre (*then* ve *else*) herhangi bir türde olabilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
condition	Mantıksal olarak yorumlanan ifade.

Bağımsız Değişken	Açıklama
then	Herhangi bir türde olabilen ifade. *condition* koşulu True ise if fonksiyonu *then* ifadesinin değerini döndürür.
else	Herhangi bir türde olabilen ifade. *condition* koşulu False ise if fonksiyonu *else* ifadesinin değerini döndürür. Bu parametre isteğe bağlıdır. *condition*, False olursa else belirtmediyseniz NULL değeri döndürülür.

Örnek

Örnek	Sonuç
`if( Amount>= 0, 'OK', 'Alarm' )`	Bu ifade, tutarın pozitif bir sayı (0 veya daha büyük) olup olmadığını test eder ve öyleyse 'OK' döndürür. Miktar 0'dan küçükse 'Alarm' sonucu döndürülür.

Örnek - if kullanan kod ekleme

Örnek: Komut dosyası

Yükleme kodu

If, değişkenler de dahil olmak üzere diğer yöntemleri ve nesneleri içeren yükleme kodunda kullanılabilir. Örneğin, bir *threshold* değişkenini ayarlar ve bu eşiğe göre veri modeline bir alanı dahil etmek isterseniz aşağıdakileri yapabilirsiniz.

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Sonuçları görmek için Qlik Sense uygulamasında tabloyu oluşturun.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753,
20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756,
20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   set
threshold = 100;                                        /* Create new table called Transaction_Buckets  Comp
amount field from Transaction table to threshold of 100. Output results into a new field
called Compared to Threshold                                                          */
If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)') as
[Compared to Threshold] Resident Transactions;
```

Sonuçlar

Yükleme kodundaki *if* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

transaction_id	Eşik ile Karşılaştırılan
3750	100'den küçük
3751	100'den büyük
3752	100'den küçük
3753	100'den büyük
3754	100'den büyük
3756	100'den küçük
3757	100'den büyük

Örnekler - if kullanan grafik ifadeleri

Örnekler: Grafik ifadeleri

Grafik ifadesi 1

Komut dosyası

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Verileri yükledikten sonra, bir Qlik Sense tablosunda aşağıdaki grafik ifadesi örneklerini oluşturun.

```
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];
```

Bir grafik ifadesinde *if* fonksiyonunun örneklerini gösteren Qlik Sense tablosu.

Tarih	Konum	Olaylar	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Beijing	0	Tamam	Tamam
1/3/2016	Boston	12	Kritik	Kritik
1/3/2016	Stockholm	3	Tamam	Uyarı
1/3/2016	Toronto	0	Tamam	Tamam
1/4/2016	Beijing	0	Tamam	Tamam
1/4/2016	Boston	8	Tamam	Uyarı

Grafik ifadesi 2

Yeni bir uygulamada, aşağıdaki kodu veri yükleme düzenleyicisinde yeni bir sekmede ekleyin, sonra verileri yükleyin. Daha sonra tabloyu aşağıdaki grafik ifadeleriyle oluşturabilirsiniz.

```
SET FirstWeekDay=0; Load  Date(MakeDate(2022)+RecNo()-1) as Date Autogenerate 14;
```

Bir grafik ifadesinde *if* fonksiyonunun bir örneğini gösteren Qlik Sense tablosu.

Tarih	WeekDay(Date)	If(WeekDay (Date)>=5,'Hafta Sonu','Normal Gün')
1/1/2022	Cmt	Hafta Sonu
1/2/2022	Paz	Hafta Sonu
1/3/2022	Pzt	Normal Gün
1/4/2022	Sal	Normal Gün
1/5/2022	Çar	Normal Gün
1/6/2022	Per	Normal Gün
1/7/2022	Cum	Normal Gün
1/8/2022	Cmt	Hafta Sonu
1/9/2022	Paz	Hafta Sonu
1/10/2022	Pzt	Normal Gün
1/11/2022	Sal	Normal Gün
1/12/2022	Çar	Normal Gün
1/13/2022	Per	Normal Gün
1/14/2022	Cum	Normal Gün

match

match fonksiyonu, ilk parametreyi ondan sonra gelen tüm parametrelerle karşılaştırır ve eşleşen ifadelerin sayısal konumunu döndürür. Karşılaştırma büyük/küçük harf duyarlıdır.

Söz Dizimi:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Büyük/Küçük harf duyarlılığı olmayan karşılaştırma kullanmak isterseniz **mixmatch** fonksiyonunu kullanın. Büyük/Küçük harf duyarlılığı olan karşılaştırma ve joker karakterler kullanmak isterseniz **wildmatch** fonksiyonunu kullanın.*

Örnek: match kullanan yükleme kodu

Örnek: Yükleme kodu

Yükleme kodu

Veri alt kümesini yüklemek için match öğesini kullanabilirsiniz. Örneğin, fonksiyondaki bir ifade için sayısal değer döndürebilirsiniz. Daha sonra sayısal değere göre verileri sınırlayabilirsiniz. Bir eşleşme olmadığında Match, 0 değerini döndürür. Bu nedenle bu örnekte eşleşmeyen tüm ifadeler 0 değerini döndürür ve WHERE deyimi tarafından veri yüklemesinden hariç tutulur.

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Sonuçları görmek için Qlik Sense uygulamasında tabloyu oluşturun.

```
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;
```

Sonuçlar

Yükleme kodundaki match fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu

Renk Kodu Blue ve Black	Customer
Black	203521
Black	3036491
Blue	2038593

Örnekler - match kullanan grafik ifadeleri

Örnekler: Grafik ifadeleri

Grafik ifadesi 1

Komut dosyası

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Verileri yükledikten sonra, bir Qlik Sense tablosunda aşağıdaki grafik ifadesi örneklerini oluşturun.

```
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];
```

'Stockholm', **match** fonksiyonundaki ifade listesinde yer almadığından, aşağıdaki tablodaki ilk ifade, Stockholm için 0 değerini döndürür. Ayrıca **match** karşılaştırması büyük/küçük harf duyarlı olduğundan 'Zurich' için de 0 değerini döndürür.

Bir grafik ifadesinde *match* fonksiyonunun örneklerini gösteren Qlik Sense tablosu

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

Grafik ifadesi 2

Bir ifade için özel bir sıralama gerçekleştirmek için match öğesini kullanabilirsiniz.

Varsayılan olarak sütunlar, verilere bağlı olarak sayısal ve alfabetik şekilde sıralanır.

Varsayılan sıralama düzeni örneğini gösteren Qlik Sense tablosu

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Sıralamayı değiştirmek için aşağıdakileri yapın:

1. **Özellikler** panelinde grafiğiniz için **Sıralama** bölümünü açın.
2. Özel sıralama yapmak istediğiniz sütun için otomatik sıralamayı kapatın.
3. **Sayısal olarak sırala** ve **Alfabetik olarak sırala** seçeneğinin seçimini kaldırın.
4. **İfadeye göre sırala** seçeneğini belirleyin ve şuna benzer bir ifade girin:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Cities sütunundaki sıralama düzeni değişir.

match fonksiyonunu kullanarak sıralama düzenini değiştirme örneğini gösteren Qlik Sense tablosu

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Döndürülen sayısal değeri de görüntüleyebilirsiniz.

match fonksiyonundan döndürülen sayısal değerlerin örneğini gösteren Qlik Sense tablosu

Şehirler	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

mixmatch fonksiyonu, ilk parametreyi ondan sonra gelen tüm parametrelerle karşılaştırır ve eşleşen ifadelerin sayısal konumunu döndürür. Karşılaştırma büyük/küçük harf duyarlı değildir.

Söz Dizimi:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Bunun yerine büyük/küçük harfe duyarlı karşılaştırma kullanmak istiyorsanız, **match** fonksiyonunu kullanın. Büyük/Küçük harf duyarlılığı olan karşılaştırma ve joker karakterler kullanmak isterseniz **wildmatch** fonksiyonunu kullanın.

Örnek - mixmatch kullanan kod ekleme

Örnek: Yükleme kodu

Yükleme kodu

Veri alt kümesini yüklemek için mixmatch öğesini kullanabilirsiniz. Örneğin, fonksiyondaki bir ifade için sayısal değer döndürebilirsiniz. Daha sonra sayısal değere göre verileri sınırlayabilirsiniz. Bir eşleşme olmadığında Mixmatch, 0 değerini döndürür. Bu nedenle bu örnekte eşleşmeyen tüm ifadeler 0 değerini döndürür ve WHERE deyimi tarafından veri yüklemesinden hariç tutulur.

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Sonuçları görmek için Qlik Sense uygulamasında tabloyu oluşturun.

```
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;
```

Sonuçlar

Yükleme kodundaki mixmatch fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

Renk kodu Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Örnekler - mixmatch kullanan grafik ifadeleri

Örnekler: Grafik ifadeleri

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Verileri yükledikten sonra, bir Qlik Sense tablosunda aşağıdaki grafik ifadesi örneklerini oluşturun.

Grafik ifadesi 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];
```

'Stockholm', **mixmatch** fonksiyonundaki ifade listesinde yer almadığından, aşağıdaki tablodaki ilk ifade, Stockholm için 0 değerini döndürür. **mixmatch** karşılaştırması büyük/küçük harf duyarlı olmadığından 'Zurich' için 4 değerini döndürür.

Bir grafik ifadesinde *mixmatch* fonksiyonunun örneklerini gösteren Qlik Sense tablosu

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

Grafik ifadesi 2

Bir ifadede özel bir sıralama işlemi yapmak için mixmatch kullanabilirsiniz.

Varsayılan olarak sütunlar, verilere bağlı olarak alfabetik veya sayısal şekilde sıralanır.

Varsayılan sıralama düzeni örneğini gösteren Qlik Sense tablosu

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Sıralamayı değiştirmek için aşağıdakileri yapın:

1. **Özellikler** panelinde grafiğiniz için **Sıralama** bölümünü açın.
2. Özel sıralama yapmak istediğiniz sütun için otomatik sıralamayı kapatın.
3. **Sayısal olarak sırala** ve **Alfabetik olarak sırala** seçeneğinin seçimini kaldırın.
4. **İfadeye göre sırala**'yı seçin, sonra aşağıdaki ifadeyi girin:
   ```
   =mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')
   ```
   Cities sütunundaki sıralama düzeni değişir.

mixmatch fonksiyonunu kullanarak sıralama düzenini değiştirme örneğini gösteren Qlik Sense tablosu.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Döndürülen sayısal değeri de görüntüleyebilirsiniz.

mixmatch fonksiyonundan döndürülen sayısal değerlerin örneğini gösteren Qlik Sense tablosu.

Şehirler	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

Pick fonksiyonu listedeki *n*. ifadeyi döndürür.

Söz Dizimi:

```
pick(n, expr1[ , expr2,...exprN])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n	*n*, 1 ile N arasında bir tamsayıdır.

Örnek:

Örnek

Örnek	Sonuç
`pick( N, 'A','B',4, 6 )`	N = 2 ise 'B' döndürür N = 3 ise 4 döndürür

wildmatch

wildmatch fonksiyonu ilk parametreyi sonraki tüm parametrelerle karşılaştırır ve eşleşen ifadenin sayısını döndürür. Karşılaştırma dizelerinde joker karakterlerin (***** ve **?**) kullanılmasına izin verir. *****, herhangi bir karakter sırasını eşleştirir. **?**, herhangi bir tek karakterle eşleşir. Karşılaştırma büyük/küçük harf duyarlı değildir.

Söz Dizimi:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Joker karakterler olmayan karşılaştırma kullanmak isterseniz **match** veya **mixmatch** fonksiyonlarını kullanın.

Örnek: wildmatch kullanan yükleme kodu

Örnek: Yükleme kodu

Yükleme kodu

Veri alt kümesini yüklemek için wildmatch öğesini kullanabilirsiniz. Örneğin, fonksiyondaki bir ifade için sayısal değer döndürebilirsiniz. Daha sonra sayısal değere göre verileri sınırlayabilirsiniz. Bir eşleşme olmadığında Wildmatch, 0 değerini döndürür. Bu nedenle bu örnekte eşleşmeyen tüm ifadeler 0 değerini döndürür ve WHERE deyimi tarafından veri yüklemesinden hariç tutulur.

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Sonuçları görmek için Qlik Sense uygulamasında tabloyu oluşturun.

```
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;
```

Sonuçlar

Yükleme kodundaki *wildmatch* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu

Renk Kodu Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Örnekler: wildmatch kullanan grafik ifadeleri

Örnek: Grafik ifadesi

Grafik ifadesi 1

Veri yükleme düzenleyicisinde yeni bir sekme oluşturun, sonra aşağıdaki verileri satır içi yükleme olarak yükleyin. Verileri yükledikten sonra, bir Qlik Sense tablosunda aşağıdaki grafik ifadesi örneklerini oluşturun.

```
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];
```

'Stockholm', **wildmatch** fonksiyonundaki ifade listesinde yer almadığından, aşağıdaki tablodaki ilk ifade, Stockholm için 0 değerini döndürür. Ayrıca ? yalnızca tek bir karakterle eşleştiğinden 'Boston' için de 0 değerini döndürür.

Bir grafik ifadesinde *wildmatch* fonksiyonunun örneklerini gösteren Qlik Sense tablosu

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

Grafik ifadesi 2

Bir ifadede özel bir sıralama işlemi yapmak için wildmatch kullanabilirsiniz.

Varsayılan olarak sütunlar, verilere bağlı olarak sayısal ve alfabetik şekilde sıralanır.

Varsayılan sıralama düzeni örneğini gösteren Qlik Sense tablosu

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Sıralamayı değiştirmek için aşağıdakileri yapın:

1. **Özellikler** panelinde grafiğiniz için **Sıralama** bölümünü açın.
2. Özel sıralama yapmak istediğiniz sütun için otomatik sıralamayı kapatın.
3. **Sayısal olarak sırala** ve **Alfabetik olarak sırala** seçeneğinin seçimini kaldırın.
4. **İfadeye göre sırala** seçeneğini belirleyin ve şuna benzer bir ifade girin:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Cities sütunundaki sıralama düzeni değişir.

wildmatch fonksiyonunu kullanarak sıralama düzenini değiştirme örneğini gösteren Qlik Sense tablosu.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Döndürülen sayısal değeri de görüntüleyebilirsiniz.

wildmatch fonksiyonundan döndürülen sayısal değerlerin örneğini gösteren Qlik Sense tablosu

Şehirler	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Sayaç işlevleri

Bu bölümde, veri kod dosyasındaki **LOAD** deyiminin değerlendirilmesi sırasında kayıt sayaçları ile ilgili fonksiyonlar açıklanmaktadır. Grafik ifadelerinde kullanılabilecek tek fonksiyon **RowNo()** fonksiyonudur.

Bazı sayaç işlevlerinin parametresi yoktur; ancak sondaki parantezler yine de gereklidir.

Sayaç işlevlerine genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

autonumber
Kod fonksiyonu, kod yürütme sırasında karşılaşılan *expression* öğesinin her tekil değerlendirilen değeri için benzersiz bir tamsayı değerini döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Bu kod fonksiyonu, birleştirilen giriş ifadesi değerlerinin 128 bit karmasını hesaplar ve kod yürütme sırasında karşılaşılan her tekil karma değeri için benzersiz bir tamsayı değeri döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Bu kod fonksiyonu, birleştirilen giriş ifadesi değerlerinin 256 bit karmasını hesaplar ve kod yürütme sırasında karşılaşılan her tekil karma değeri için benzersiz bir tamsayı değeri döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

```
autonumberhash256 (expression {, expression})
```

IterNo

Bu kod fonksiyonu, tek kaydın bir **while** cümlesiyle bir **LOAD** deyiminde değerlendirildiği zamanı gösteren bir tamsayı döndürür. İlk yinelemenin sayısı 1'dir. **IterNo** fonksiyonu yalnızca bir **while** cümlesiyle birlikte kullanılırsa anlamlıdır.

```
IterNo ( )
```

RecNo

Bu kod fonksiyonları, dahili tablonun geçerli olarak okunan satırının sayısı için bir tamsayı döndürür. İlk kaydın sayısı 1'dir.

```
RecNo ( )
```

RowNo - script function

Bu fonksiyon, sonuç olarak elde edilen Qlik Sense dahili tablosundaki geçerli satırın konumu için bir tamsayı döndürür. İlk satırın sayısı 1'dir.

```
RowNo ( )
```

RowNo - chart function

RowNo(), bir tablodaki geçerli sütun segmentinde bulunan geçerli satırın numarasını döndürür. Bit eşlem grafikleri için **RowNo()**, grafiğin düz tablo eşdeğerindeki geçerli satırın numarasını döndürür.

```
RowNo - grafik fonksiyonu([TOTAL])
```

autonumber

Kod fonksiyonu, kod yürütme sırasında karşılaşılan *expression* öğesinin her tekil değerlendirilen değeri için benzersiz bir tamsayı değerini döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

*Tamsayı, tablonun okunduğu sıraya göre oluşturulduğundan, yalnızca aynı veri yüklemesinde oluşturulmuş **autonumber** anahtarlarını bağlayabilirsiniz. Kaynak veri sıralamasından bağımsız olarak, veri yükleri arasında kalıcı olan anahtarları kullanmanız gerekirse, **hash128**, **hash160** veya **hash256** fonksiyonlarını kullanmalısınız.*

Söz Dizimi:

```
autonumber(expression[ , AutoID])
```

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
AutoID	**autonumber** fonksiyonunun kod içindeki farklı anahtarlarda kullanılması durumunda çoklu sayaç örnekleri oluşturmak için, her bir sayacı adlandırmak üzere isteğe bağlı *AutoID* parametresi kullanılabilir.

Örnek: Bileşik anahtar oluşturma

Bu örnekte, belleği muhafaza etmek için **autonumber** fonksiyonunu kullanarak bir bileşik anahtar oluşturuyoruz. Örnek, gösterim amacına yönelik olarak kısadır; ancak çok sayıda satır içeren bir tablo ile anlamlı olur.

Örnek veriler

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

Kaynak veriler, satır içi verilerin kullanımıyla yüklenir. Daha sonra Region, Year ve Month alanlarından bileşik anahtar oluşturan bir öncelikli yüklemeyi ekliyoruz.

```
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
];
```

Elde edilen tablo şöyle görünür:

Sonuçlar tablosu

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

Bu örnekte, başka bir tabloya bağlamanız gerekmesi halinde 'North2014May' dizesi yerine RYMkey öğesine (örneğin, 1) referansta bulunabilirsiniz.

Şimdi de maliyetleri içeren bir kaynak tabloyu benzer şekilde yüklüyoruz. Yapay anahtar oluşturmanın önüne geçmek için Region, Year ve Month alanları öncelikli yüklemede hariç tutulur; tabloları bağlayarak **autonumber** fonksiyonu ile bir bileşik anahtar zaten oluşturuyoruz.

```
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
];
```

Artık bir sayfaya bir tablo görselleştirmesi ekleyebilir ve Region, Year ve Month alanlarının yanı sıra satış ve maliyetlere ilişkin Toplam hesaplamalarını ilave edebiliriz. Tablo şöyle görünür:

Sonuçlar tablosu

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

Bu kod fonksiyonu, birleştirilen giriş ifadesi değerlerinin 128 bit karmasını hesaplar ve kod yürütme sırasında karşılaşılan her tekil karma değeri için benzersiz bir tamsayı değeri döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

*Tamsayı, tablonun okunduğu sıraya göre oluşturulduğundan, yalnızca aynı veri yüklemesinde oluşturulmuş **autonumberhash128** anahtarlarını bağlayabilirsiniz. Kaynak veri sıralamasından bağımsız olarak, veri yükleri arasında kalıcı olan anahtarları kullanmanız gerekirse, **hash128**, **hash160** veya **hash256** fonksiyonlarını kullanmalısınız.*

Söz Dizimi:

```
autonumberhash128(expression {, expression})
```

Örnek: Bileşik anahtar oluşturma

Bu örnekte, belleği muhafaza etmek için **autonumberhash128** fonksiyonunu kullanarak bir bileşik anahtar oluşturuyoruz. Örnek, gösterim amacına yönelik olarak kısadır; ancak çok sayıda satır içeren bir tablo ile anlamlı olur.

Örnek veriler

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

Kaynak veriler, satır içi verilerin kullanımıyla yüklenir. Daha sonra Region, Year ve Month alanlarından bileşik anahtar oluşturan bir öncelikli yüklemeyi ekliyoruz.

```
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
];
```

Elde edilen tablo şöyle görünür:

Sonuçlar tablosu

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

Bu örnekte, başka bir tabloya bağlamanız gerekmesi halinde 'North2014May' dizesi yerine RYMkey öğesine (örneğin, 1) referansta bulunabilirsiniz.

Şimdi de maliyetleri içeren bir kaynak tabloyu benzer şekilde yüklüyoruz. Yapay anahtar oluşturmanın önüne geçmek için Region, Year ve Month alanları öncelikli yüklemede hariç tutulur; tabloları bağlayarak **autonumberhash128** fonksiyonu ile bir bileşik anahtar zaten oluşturuyoruz.

```
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
];
```

Artık bir sayfaya bir tablo görselleştirmesi ekleyebilir ve Region, Year ve Month alanlarının yanı sıra satış ve maliyetlere ilişkin Toplam hesaplamalarını ilave edebiliriz. Tablo şöyle görünür:

Sonuçlar tablosu

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56

Region	Year	Month	Sum([Sales])	Sum([Costs])
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Bu kod fonksiyonu, birleştirilen giriş ifadesi değerlerinin 256 bit karmasını hesaplar ve kod yürütme sırasında karşılaşılan her tekil karma değeri için benzersiz bir tamsayı değeri döndürür. Bu fonksiyon, örneğin bir karmaşık anahtarın sıkıştırılmış bellek temsilini oluşturmak için kullanılabilir.

*Tamsayı, tablonun okunduğu sıraya göre oluşturulduğundan, yalnızca aynı veri yüklemesinde oluşturulmuş **autonumberhash256** anahtarlarını bağlayabilirsiniz. Kaynak veri sıralamasından bağımsız olarak, veri yükleri arasında kalıcı olan anahtarları kullanmanız gerekirse, **hash128**, **hash160** veya **hash256** fonksiyonlarını kullanmalısınız.*

Söz Dizimi:

```
autonumberhash256(expression {, expression})
```

Örnek: Bileşik anahtar oluşturma

Bu örnekte, belleği muhafaza etmek için **autonumberhash256** fonksiyonunu kullanarak bir bileşik anahtar oluşturuyoruz. Örnek, gösterim amacına yönelik olarak kısadır; ancak çok sayıda satır içeren bir tablo ile anlamlı olur.

Örnek tablo

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

Kaynak veriler, satır içi verilerin kullanımıyla yüklenir. Daha sonra Region, Year ve Month alanlarından bileşik anahtar oluşturan bir öncelikli yüklemeyi ekliyoruz.

```
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
];
```

Elde edilen tablo şöyle görünür:

Sonuçlar tablosu

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

Bu örnekte, başka bir tabloya bağlamanız gerekmesi halinde 'North2014May' dizesi yerine RYMkey öğesine (örneğin, 1) referansta bulunabilirsiniz.

Şimdi de maliyetleri içeren bir kaynak tabloyu benzer şekilde yüklüyoruz. Yapay anahtar oluşturmanın önüne geçmek için Region, Year ve Month alanları öncelikli yüklemede hariç tutulur; tabloları bağlayarak **autonumberhash256** fonksiyonu ile bir bileşik anahtar zaten oluşturuyoruz.

```
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
];
```

Artık bir sayfaya bir tablo görselleştirmesi ekleyebilir ve Region, Year ve Month alanlarının yanı sıra satış ve maliyetlere ilişkin Toplam hesaplamalarını ilave edebiliriz. Tablo şöyle görünür:

Sonuçlar tablosu

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

Bu kod fonksiyonu, tek kaydın bir **while** cümlesiyle bir **LOAD** deyiminde değerlendirildiği zamanı gösteren bir tamsayı döndürür. İlk yinelemenin sayısı 1'dir. **IterNo** fonksiyonu yalnızca bir **while** cümlesiyle birlikte kullanılırsa anlamlıdır.

Söz Dizimi:

```
IterNo( )
```

Örnekler ve sonuçlar:

Örnek:

```
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
];
```

Bu **LOAD** deyimi, **StartDate** ve **EndDate** ile tanımlanan aralık dahilinde her tarih için bir kayıt oluşturur.

Bu örnekte, sonuçta elde edilen tablo şuna benzer:

Sonuçlar tablosu

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Bu kod fonksiyonları, dahili tablonun geçerli olarak okunan satırının sayısı için bir tamsayı döndürür. İlk kaydın sayısı 1'dir.

Söz Dizimi:

```
RecNo( )
```

Sonuçta elde edilen Qlik Sense tablosundaki satırları sayan **RowNo( )** fonksiyonunun aksine, **RecNo( )** fonksiyonu ham veri tablosundaki kayıtları sayar ve ham veri tablosu bir diğeriyle birleştirildiğinde sıfırlanır.

Örnek: Veri kod dosyası

Ham veri tablosu yüklemesi:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Seçilen satırlar için kayıt ve satır sayılarını yükleme:

```
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;
```

Sonuç olarak elde edilen Qlik Sense dahili tablosu:

Sonuçlar tablosu

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Bu fonksiyon, sonuç olarak elde edilen Qlik Sense dahili tablosundaki geçerli satırın konumu için bir tamsayı döndürür. İlk satırın sayısı 1'dir.

Söz Dizimi:

```
RowNo( [TOTAL])
```

Ham veri tablosundaki kayıtları sayan **RecNo( )** fonksiyonunun aksine, **RowNo( )** fonksiyonu **where** cümlelerinin hariç tuttuğu kayıtları saymaz ve ham veri tablosu bir diğeriyle birleştirildiğinde sıfırlanmaz.

> *Öncelikli yüklemeyi, yani aynı tablodan okuma yapan bir dizi yığınlanmış **LOAD** deyimini kullanıyorsanız, **RowNo( )** fonksiyonunu yalnızca en üst **LOAD** deyiminde kullanabilirsiniz. **RowNo( )** fonksiyonunu sonraki **LOAD** deyimlerinde kullanırsanız 0 sonucu döndürülür.*

Örnek: Veri kod dosyası

Ham veri tablosu yüklemesi:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Seçilen satırlar için kayıt ve satır sayılarını yükleme:

```
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;
```

Sonuç olarak elde edilen Qlik Sense dahili tablosu:

Sonuçlar tablosu

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - grafik fonksiyonu

RowNo(), bir tablodaki geçerli sütun segmentinde bulunan geçerli satırın numarasını döndürür. Bit eşlem grafikleri için **RowNo()**, grafiğin düz tablo eşdeğerindeki geçerli satırın numarasını döndürür.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sütun segmentleri

*Grafiğin ifadelerinden herhangi birinde **RowNo()** kullanıldığında, grafiklerdeki y-değerleri üzerinde sıralamaya veya tablolardaki ifade sütunlarına göre sıralamaya izin verilmez. Bu nedenle, söz konusu sıralama alternatifleri otomatik olarak devre dışı bırakılır.*

Söz Dizimi:

```
RowNo([TOTAL])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Örnek: RowNo kullanan grafik ifadesi

Örnek - grafik ifadesi

Komut dosyası

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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
'|');
```

Grafik ifadesi

Qlik Sense sayfasında **Customer** ve **UnitSales** alanlarını boyut olarak kullanarak bir tablo görselleştirmesi oluşturun. `RowNo( )` ve `RowNo(TOTAL)` işlevlerini, sırasıyla **Segmentteki Satır** ve **Row Number** olarak etiketleyerek hesaplama olarak ekleyin. Tabloya hesaplama olarak şu ifadeyi ekleyin:

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Sonuç

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

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
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

Açıklama

Row in Segment sütunu, Astrida müşterisine ait UnitSales değerlerini içeren sütun segmenti için 1,2,3 sonuçlarını gösterir. Daha sonra satır numaralandırması bir sonraki sütun segmenti (yani, Betacab) için tekrar 1'den başlar.

Row Number sütunu, `RowNo()` için `TOTAL` bağımsız değişkeni nedeniyle boyutları yoksayar ve tablodaki satırları sayar.

İfade, her sütun dilimindeki ilk satır için 0 döndürür, bu nedenle sütun şunu gösterir:

0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2, 0 ve 4.

Ayrıca bkz.

Above - grafik fonksiyonu (page 651)

5.7 Tarih ve saat fonksiyonları

Qlik Sense tarih ve saat fonksiyonları, tarih ve saat değerlerini dönüştürmek için kullanılır. Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

Fonksiyonlar, 30 Aralık 1899'dan beri gün sayısına eşit olan tarih-saat seri numarasını temel alır. Tamsayı değeri günü ve kesir değeri günün saatini temsil eder.

Qlik Sense parametrenin sayısal değerini kullandığından, bir sayı tarih veya saat olarak biçimlendirilmemiş olsa bile parametre olarak geçerlidir. Parametre, örneğin bir dize olması nedeniyle sayısal değere karşılık gelmiyorsa, Qlik Sense bu dizeyi tarih ve saat ortam değişkenlerine göre yorumlamaya çalışır.

Parametrede kullanılan saat biçimi ortam değişkenlerinde ayarlanan biçime karşılık gelmiyorsa, Qlik Sense doğru bir yorumlama yapamaz. Bu sorunu çözmek için ayarları değiştirin veya bir yorumlama fonksiyonunu kullanın.

Her bir fonksiyona ilişkin örneklerde, varsayılan saat ve tarih biçimlerinin hh:mm:ss ve YYYY-MM-DD (ISO 8601) olduğu varsayılmaktadır.

Qlik Sense, tarih veya zaman fonksiyonu olan bir zaman damgasını işlerken tarih veya zaman fonksiyonu bir coğrafi konum içermediği sürece yaz saati parametrelerini yoksayar.

Örneğin, `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` tarafından yaz saati parametreleri kullanılırken `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` tarafından kullanılmaz.

Tarih ve saat fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Tamsayı zaman ifadeleri

second

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, saniyeyi temsil eden bir tamsayı döndürür.

```
second (expression)
```

minute

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, dakikayı temsil eden bir tamsayı döndürür.

```
minute (expression)
```

hour

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, saati temsil eden bir tamsayı döndürür.

```
hour (expression)
```

day

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre tarih olarak yorumlandığında, günü temsil eden bir tamsayı döndürür.

```
day (expression)
```

week

Bu fonksiyon, ISO 8601 uyarınca hafta numarasını temsil eden bir tamsayı döndürür. Hafta numarası, standart sayı yorumlamasına göre ifadenin tarih yorumlamasından hesaplanır.

```
week (expression)
```

month

Bu fonksiyon, ikili değer döndürür: **MonthNames** ortam değişkeninde tanımlandığı şekliyle ay adı ve 1-12 arasında bir tamsayı. Ay, standart sayı yorumlamasına göre ifadenin tarih yorumlamasından hesaplanır.

```
month (expression)
```

year

Bu fonksiyon, **expression** öğesi standart sayı yorumlamasına göre tarih olarak yorumlandığında, yılı temsil eden bir tamsayı döndürür.

```
year (expression)
```

weekyear

Bu fonksiyon, ISO 8601 uyarınca hafta numarasının ait olduğu yılı döndürür. Hafta sayısı, 1 ve yaklaşık 52 arasında değişir.

```
weekyear (expression)
```

weekday

Bu fonksiyon şunları içeren bir ikili değer döndürür: **DayNames** ortam değişkeninde tanımlanan bir gün adı. Haftanın nominal gününe karşılık gelen 0-6 arasında bir tamsayı (0-6).

```
weekday (date)
```

Zaman damgası fonksiyonları

now

Bu fonksiyon, sistem saatinden geçerli zamanın zaman damgasını döndürür. Varsayılan değer 1'dir.

```
now ([ timer_mode])
```

today

Bu fonksiyon, sistem saatinden geçerli tarihi döndürür.

```
today ([timer_mode])
```

LocalTime

Bu fonksiyon, belirtilen bir saat dilimi için sistem saatinden geçerli zamanın zaman damgasını döndürür.

```
localtime ([timezone [, ignoreDST ]])
```

"Make" fonksiyonları

makedate

Bu fonksiyon **YYYY** yılı, **MM** ayı ve **DD** gününden hesaplanan bir tarih döndürür.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Bu fonksiyon **YYYY** yılı, **WW** haftası ve **D** haftanın gününden hesaplanan bir tarih döndürür.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Bu fonksiyon **hh** saati, **mm** dakikası ve **ss** saniyesinden hesaplanan bir saat döndürür.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Diğer tarih fonksiyonları

AddMonths

Fonksiyon, **startdate** değerinden **n** ay sonra olan tarihi veya **n** negatif ise, **startdate** değerinden **n** ay önce olan tarihi döndürür.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Fonksiyon, **startdate** değerinden **n** yıl sonra olan tarihi veya **n** negatif ise, **startdate** değerinden **n** yıl önce olan tarihi döndürür.

```
addyears (startdate, n)
```

yeartodate

Bu fonksiyon giriş zaman damgasının kodun yüklendiği yılda olup olmadığını bulur ve bu yıldaysa True, değilse False değerini döndürür.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Saat dilimi fonksiyonları

timezone

Bu fonksiyon, Windows'da tanımlandığı şekilde, geçerli saat diliminin adını döndürür.

```
timezone ( )
```

GMT

Bu fonksiyon, sistem saatinden ve Windows saat ayarlarından türetilen geçerli Greenwich Mean Time değerini döndürür.

```
GMT ( )
```

UTC

Geçerli Coordinated Universal Time değerini döndürür.

```
UTC ( )
```

daylightsaving

Windows'ta tanımlandığı şekilde, günışığından yararlanma saati için geçerli ayarı döndürür.

```
daylightsaving ( )
```

converttolocaltime

Bir UTC veya GMT zaman damgasını ikili değer olarak yerel zamana dönüştürür. Yer, dünyadaki bir dizi şehir, yer ve saat diliminden herhangi biri olabilir.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Zaman ayarlama fonksiyonları

setdateyear

Bu fonksiyon, giriş olarak bir **timestamp** ve **year** alır ve **timestamp** öğesini girişte belirtilen **year** ile günceller.

```
setdateyear (timestamp, year)
```

setdateyearmonth

Bu fonksiyon, giriş olarak bir **timestamp**, **month** ve **year** alır ve **timestamp** öğesini girişte belirtilen **year** ve **month** ile günceller.

```
setdateyearmonth (timestamp, year, month)
```

"In..." fonksiyonları

inyear

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren yıl içinde olması halinde True döndürür.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Bu fonksiyon, **timestamp**, yılın **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren çeyrek içinde olması halinde True döndürür.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Bu fonksiyon, **timestamp** öğesi çeyreğin **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren ay içinde olması halinde True döndürür.

```
inmonth (date, basedate , shift)
```

inmonthtodate

date, ayın **basedate** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **basedate** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

```
inmonthtodate (date, basedate , shift)
```

inmonths

Bu fonksiyon, bir zaman damgasının taban tarih ile aynı ay, iki aylık dönem, çeyrek, tersiyel veya yarım yıl içine denk gelip gelmediğini bulur. Zaman damgasının önceki veya sonraki bir zaman dönemine denk gelip gelmediğini bulmak da mümkündür.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Bu fonksiyon, bir zaman damgasının ayın, iki ayın, üç ayın, dört ayın veya altı ayın parçası ve **base_date**'in son milisaniyesi içinde olup olmadığını bulur. Zaman damgasının önceki veya sonraki bir zaman dönemine denk gelip gelmediğini bulmak da mümkündür.

```
inmonthstodate  (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren hafta içinde olması halinde True döndürür.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Bu fonksiyon, **timestamp**, haftanın **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

Bu fonksiyon, **timestamp**'ın **base_date**'i içeren ay haftasında olup olmadığını bulur. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

Bu fonksiyon, **timestamp**'ın ay haftası ile **base_date**'in son milisaniyesi arasında yer alıp almadığını bulur. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

Bu fonksiyon, **timestamp** değerinin **base_timestamp** değerini içeren gün içinde olması halinde True döndürür.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Bu fonksiyon, **timestamp** öğesi günün **base_timestamp** öğesinin tam milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_timestamp** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

"Start ... end" fonksiyonları

yearstart

Bu fonksiyon, **date** içeren yılın ilk gününün başlangıcına karşılık gelen bir zaman damgası döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Bu fonksiyon, **date** içeren yılın son gününün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Bu fonksiyon, **date** öğesini içeren yılın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle görüntü değeri olarak dört basamaklı bir yıl döndürür.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Bu fonksiyon, **date** içeren çeyreğin ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Bu fonksiyon, **date** içeren çeyreğin son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Bu fonksiyon, çeyreğin aylarını (**MonthNames** kod değişkenine göre biçimlendirilmiş) ve yılı, çeyreğin ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle gösteren bir görüntü değeri döndürür.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Bu fonksiyon, **date** içeren ayın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
monthstart (date [, shift = 0])
```

monthend

Bu fonksiyon, **date** içeren ayın son gününün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
monthend (date [, shift = 0])
```

monthname

Bu fonksiyon, ayı (**MonthNames** kod değişkenine göre biçimlendirilmiş) ve yılı, ayın ilk gününün ilk milisaniyesine sahip zaman damgasına karşılık gelen bir temel sayısal değerle gösteren bir görüntü değeri döndürür.

```
monthname (date [, shift = 0])
```

monthsstart

Bu fonksiyon bir taban tarih içeren ayın, iki aylık dönemin, çeyreğin, tersiyelin veya yarım yılın ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Önceki ve sonraki bir zaman dönemi için zaman damgasını bulmak da mümkündür.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Bu fonksiyon bir taban tarih içeren ayın, iki aylık dönemin, çeyreğin, tersiyelin veya yarım yılın son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Önceki ve sonraki bir zaman dönemi için zaman damgasını bulmak da mümkündür.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Bu fonksiyon, dönemin ay aralığının (**MonthNames** kod değişkenine göre biçimlendirilmiş) yanı sıra yılı temsil eden bir görüntü değeri döndürür. Temel sayısal değer; ayın, iki ayın, üç ayın, dört ayın veya altı ayın temel tarih içeren ilk milisaniyesinin zaman damgasına karşılık gelir.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Bu fonksiyon, **date** içeren takvim haftasının ilk gününün (Pazartesi) ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Bu fonksiyon, **date** öğesini içeren takvim haftasının son günün (Pazar) son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi, kodda ayarlanan **DateFormat** olur.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Bu fonksiyon, **date** öğesini içeren haftanın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle yıl ve hafta sayısını gösteren bir değer döndürür.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Bu fonksiyon, **date** içeren ay haftasının ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Bu fonksiyon, **date** içeren ay haftasının son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Bu fonksiyon, **date** içeren ay haftasının ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen yıl ve ay haftası numarasını gösteren bir görüntü değeri döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Bu fonksiyon, **time** bağımsız değişkenindeki günün ilk milisaniyesini içeren bir zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **TimestampFormat** olur.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Bu fonksiyon, **time** içindeki günün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **TimestampFormat** olur.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Bu fonksiyon, **time** öğesini içeren günün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle tarihi gösteren bir değer döndürür.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Gün numaralandırma fonksiyonları

age

age fonksiyonu, **date_of_birth** tarihinde doğan birinin **timestamp** sırasındaki yaşını (tamamlanan yıl cinsinden) döndürür.

```
age (timestamp, date_of_birth)
```

networkdays

networkdays fonksiyonu, isteğe bağlı olarak listelenen tüm **holiday** öğelerini dikkate alarak, **start_date** ve **end_date** arasındaki ve bu tarihleri de içeren iş günlerinin (Pazartesi - Cuma) sayısını döndürür.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

firstworkdate fonksiyonu, isteğe bağlı olarak listelenen tüm tatilleri dikkate alarak, **end_date** tarihinden önce biten **no_of_workdays** (Pazartesi - Cuma) değerini elde etmek için en son başlangıç tarihini döndürür. **end_date** ve **holiday** geçerli tarihler veya zaman damgaları olmalıdır.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

lastworkdate fonksiyonu, isteğe bağlı **holiday** varsa bunları da dikkate alarak, **start_date** ile başlanması halinde **no_of_workdays** (Pazartesi-Cuma) elde edilmesi için gerekli en erken bitiş tarihini döndürür. **start_date** ve **holiday** geçerli tarihler veya zaman damgaları olmalıdır.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Bu fonksiyon bir zaman damgasının denk geldiği yılın gün numarasını hesaplar. Hesaplama yılın ilk gününün ilk milisaniyesinden itibaren yapılır, ancak ilk ay kaymış olabilir.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Bu fonksiyon bir zaman damgasının denk geldiği çeyreğin gün numarasını hesaplar.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Fonksiyon, **startdate** değerinden **n** ay sonra olan tarihi veya **n** negatif ise, **startdate** değerinden **n** ay önce olan tarihi döndürür.

Söz Dizimi:

```
AddMonths(startdate, n , [ , mode])
```

Dönüş verileri türü: dual

AddMonths fonksiyonu hem dize hem de sayı değeri içeren bir ikili değer döndürür. Fonksiyon, giriş ifadesinin sayısal değerini alır ve sayıyı temsil eden bir dize oluşturur. Dize görüntülenir, ancak sayısal değer, tüm sayısal hesaplamalar ve sıralama için kullanılır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
startdate	Bir zaman damgası olarak başlangıç tarihi; örneğin '2012-10-12'.
n	Pozitif veya negatif tamsayı olarak ay sayısı.
mode	Ayın başına göre mi, yoksa sonuna göre mi ay eklendiğini belirtir. Varsayılan mod, ayın başına göre eklemeler için 0 olur. Ayın sonuna göre eklemeler için modu 1 olarak ayarlayın. Mod 1 olarak ayarlandığında ve giriş tarihi 28 veya daha sonraki bir tarih olduğunda fonksiyon, başlangıç tarihinde ayın sonuna ulaşmak için kaç gün kaldığını denetler. Döndürülen tarihte, ayın sonuna ulaşmak için aynı gün sayısı ayarlanır.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`addmonths ('2003-01-29',3)`	'2003-04-29' döndürür
`addmonths ('2003-01-29',3,0)`	'2003-04-29' döndürür
`addmonths ('2003-01-29',3,1)`	'2003-04-28' döndürür
`addmonths ('2003-01-29',1,0)`	'2003-02-28' döndürür
`addmonths ('2003-01-29',1,1)`	'2003-02-26' döndürür
`addmonths ('2003-02-28',1,0)`	'2003-03-28' döndürür
`addmonths ('2003-02-28',1,1)`	'2003-03-31' döndürür
`addmonths ('2003-01-29',-3)`	'2002-10-29' döndürür

addyears

Fonksiyon, **startdate** değerinden **n** yıl sonra olan tarihi veya **n** negatif ise, **startdate** değerinden **n** yıl önce olan tarihi döndürür.

Söz Dizimi:

```
AddYears(startdate, n)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
startdate	Bir zaman damgası olarak başlangıç tarihi; örneğin '2012-10-12'.
n	Pozitif veya negatif tamsayı olarak yıl sayısı.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`addyears ('2010-01-29',3)`	'2013-01-29' döndürür
`addyears ('2010-01-29',-1)`	'2009-01-29' döndürür

age

age fonksiyonu, **date_of_birth** tarihinde doğan birinin **timestamp** sırasındaki yaşını (tamamlanan yıl cinsinden) döndürür.

Söz Dizimi:

```
age(timestamp, date_of_birth)
```

Bir ifade olabilir.

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Tamamlanan yıl sayısının hangi zamana kadar hesaplanacağını belirten zaman damgası veya bir zaman damgasına çözümlenen ifade.
date_of_birth	Yaşı hesaplanan kişinin doğum tarihi. Bir ifade olabilir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`age('25/01/2014', '29/10/2012')`	1 döndürür.
`age('29/10/2014', '29/10/2012')`	2 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen age değerlerini gösterir.

Sonuçlar tablosu

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

Bir UTC veya GMT zaman damgasını ikili değer olarak yerel zamana dönüştürür. Yer, dünyadaki bir dizi şehir, yer ve saat diliminden herhangi biri olabilir.

Söz Dizimi:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Dönüştürülecek zaman damgası veya zaman damgasına çözümlenen ifade.

Bağımsız Değişken	Açıklama
place	Aşağıdaki geçerli yerler ve saat dilimleri tablosundan bir yer veya saat dilimi. Alternatif olarak, yerel zamanı tanımlamak için GMT veya UTC kullanabilirsiniz. Aşağıdaki değerler ve saat farkı aralıkları geçerlidir: • 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 *Yalnızca standart saat farklarını kullanabilirsiniz. Rastgele bir saat farkı (örneğin, GMT-04:27) kullanılamaz.*
ignore_dst	DST (günışığından yararlanma saati) uygulamasını göz ardı etmek istiyorsanız True olarak ayarlayın.

ignore_dst seçeneği True olarak belirlenmezse, sonuçta elde edilen saat günışığından yararlanma saati için ayarlanır.

Geçerli yerler ve saat dilimleri

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura

A-C	D-K	L-R	S-Z
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`ConvertToLocalTime('2007-11-10 23:59:00','Paris')`	'2007-11-11 00:59:00' sonucunu ve karşılık gelen dahili zaman damgası temsilini döndürür.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Kuzey Amerika doğu yakası (örneğin, New York) için saati döndürür.
`ConvertToLocalTime(UTC(), 'GMT-05:00', True)`	Kuzey Amerika doğu yakası (örneğin, New York) için saati döndürür ve günışığından yararlanma saati ayarlaması yapılmaz.

day

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre tarih olarak yorumlandığında, günü temsil eden bir tamsayı döndürür.

Fonksiyon belirli bir tarih için ayın gününü döndürür. Sıklıkla bir takvim boyutunun parçası olarak bir gün alanı türetmek için kullanılır.

Söz Dizimi:

```
day(expression)
```

Dönüş verileri türü: tamsayı

Fonksiyon örnekleri

Örnek	Sonuç
`day( '1971-10-12' )`	12 döndürür
`day( '35648' )`	35648 = 1997-08-06 olduğundan 6 döndürür

dayend

Bu fonksiyon, **time** içindeki günün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **TimestampFormat** olur.

Söz Dizimi:

```
DayEnd(time[, [period_no[, day_start]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
time	Değerlendirilecek zaman damgası.
period_no	**period_no** tamsayıya çözümlenen bir ifade olup, burada 0 değeri **time** içeren günü belirtir. **period_no** içindeki negatif değerler önceki günleri; pozitif değerler ise sonraki günleri gösterir.
day_start	Gece yarısında başlamayan günler için **day_start** içinde günün kesri olarak bir kayma belirtin. Örneğin, 0,125 değeri gece saat 3'ü belirtir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`dayend('25/01/2013 16:45:00')`	25/01/2013 23:59:59 döndürür.
`dayend('25/01/2013 16:45:00', -1)`	'24/01/2013 23:59:59 döndürür.
`dayend('25/01/2013 16:45:00', 0, 0.5)`	26/01/2013 11:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinden sonraki günün sonunu işaretleyen zaman damgasını bulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
```

```
];

InvoiceData:
LOAD *,
DayEnd(InvDate, 1) AS DEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve dayend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	DEnd
28/03/2012	29/03/2012 23:59:59
10/12/2012	11/12/2012 23:59:59
5/2/2013	07/02/2013 23:59:59
31/3/2013	01/04/2013 23:59:59
19/5/2013	20/05/2013 23:59:59
15/9/2013	16/09/2013 23:59:59
11/12/2013	12/12/2013 23:59:59
2/3/2014	03/03/2014 23:59:59
14/5/2014	15/05/2014 23:59:59
13/6/2014	14/06/2014 23:59:59
7/7/2014	08/07/2014 23:59:59
4/8/2014	05/08/2014 23:59:59

daylightsaving

Windows'ta tanımlandığı şekilde, günışığından yararlanma saati için geçerli ayarı döndürür.

Söz Dizimi:

```
DaylightSaving( )
```

Dönüş verileri türü: dual

Örnek:

```
daylightsaving( )
```

dayname

Bu fonksiyon, **time** öğesini içeren günün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle tarihi gösteren bir değer döndürür.

Söz Dizimi:

```
DayName(time[, period_no [, day_start]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
time	Değerlendirilecek zaman damgası.
period_no	**period_no** tamsayıya çözümlenen bir ifade olup, burada 0 değeri **time** içeren günü belirtir. **period_no** içindeki negatif değerler önceki günleri; pozitif değerler ise sonraki günleri gösterir.
day_start	Gece yarısında başlamayan günler için **day_start** içinde günün kesri olarak bir kayma belirtin. Örneğin, 0,125 değeri gece saat 3'ü belirtir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`dayname('25/01/2013 16:45:00')`	25/01/2013 döndürür.
`dayname('25/01/2013 16:45:00', -1)`	24/01/2013 döndürür.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	25/01/2013 döndürür. Zaman damgasının tamamı görüntülendiğinde '25/01/2013 12:00:00.000. karşılığı olan temeldeki sayısal değeri gösterir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnekte gün adı, tablodaki her bir fatura tarihinden sonraki günün başlangıcını işaretleyen zaman damgasından oluşturulur.

```
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;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve dayname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

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

Bu fonksiyon bir zaman damgasının denk geldiği çeyreğin gün numarasını hesaplar.

Söz Dizimi:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Dönüş verileri türü: tamsayı

Fonksiyon her zaman 366 günü temel alan yıllar kullanır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Değerlendirilecek tarih.
start_month	2 ile 12 arasında bir **start_month** belirtildiğinde (atlandığı takdirde 1) yılın başlangıcı herhangi bir ayın ilk gününe ileri taşınabilir. Örneğin, 1 Mart'ta başlayan bir mali yıl ile çalışmak istiyorsanız **start_month** = 3 olarak belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`DayNumberOfQuarter('12/09/2014')`	Geçerli çeyreğin gün numarası olarak 74 döndürür.
`DayNumberOfQuarter ('12/09/2014',3)`	Geçerli çeyreğin gün numarası olarak 12 döndürür. Bu örnekte ilk çeyrek Mart ile başlar (çünkü start_month 3 olarak belirtilmektedir). Bu da geçerli çeyreğin 1 Eylül'de başlayan üçüncü çeyrek olduğu anlamına gelir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfQuarter(StartDate,4) As DayNrQtr
Resident ProjectTable;
Drop table ProjectTable;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen DayNumberOfQuarter değerlerini gösterir.

Sonuçlar tablosu

InvID	StartDate	DayNrQtr
1	28/03/2014	88
2	10/12/2014	71
3	5/2/2015	36
4	31/3/2015	91
5	19/5/2015	49
6	15/9/2015	77

daynumberofyear

Bu fonksiyon bir zaman damgasının denk geldiği yılın gün numarasını hesaplar. Hesaplama yılın ilk gününün ilk milisaniyesinden itibaren yapılır, ancak ilk ay kaymış olabilir.

Söz Dizimi:

```
DayNumberOfYear(timestamp[,start_month])
```

Dönüş verileri türü: tamsayı

Fonksiyon her zaman 366 günü temel alan yıllar kullanır.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Değerlendirilecek tarih.
start_month	2 ile 12 arasında bir **start_month** belirtildiğinde (atlandığı takdirde 1) yılın başlangıcı herhangi bir ayın ilk gününe ileri taşınabilir. Örneğin, 1 Mart'ta başlayan bir mali yıl ile çalışmak istiyorsanız **start_month** = 3 olarak belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`DayNumberOfYear ('12/09/2014')`	Yılın ilk gününden itibaren sayılmasıyla gün numarası olarak 256 döndürür.
`DayNumberOfYear ('12/09/2014',3)`	1 Mart'tan itibaren sayılmasıyla günün numarası olarak 196 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfYear(StartDate,4) As DayNrYear
Resident ProjectTable;
Drop table ProjectTable;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen DayNumberOfYear değerlerini gösterir.

Sonuçlar tablosu

InvID	StartDate	DayNrYear
1	28/03/2014	363
2	10/12/2014	254
3	5/2/2015	311
4	31/3/2015	366
5	19/5/2015	49
6	15/9/2015	168

daystart

Bu fonksiyon, **time** bağımsız değişkenindeki günün ilk milisaniyesini içeren bir zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **TimestampFormat** olur.

Söz Dizimi:

```
DayStart(time[, [period_no[, day_start]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
time	Değerlendirilecek zaman damgası.
period_no	**period_no** tamsayıya çözümlenen bir ifade olup, burada 0 değeri **time** içeren günü belirtir. **period_no** içindeki negatif değerler önceki günleri; pozitif değerler ise sonraki günleri gösterir.
day_start	Gece yarısında başlamayan günler için **day_start** içinde günün kesri olarak bir kayma belirtin. Örneğin, 0,125 değeri gece saat 3'ü belirtir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`daystart('25/01/2013 16:45:00')`	25/01/2013 00:00:00 döndürür.
`daystart('25/01/2013 16:45:00', -1)`	24/01/2013 00:00:00 döndürür.
`daystart('25/01/2013 16:45:00', 0, 0.5)`	25/01/2013 12:00:00 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinden sonraki günün başlangıcını işaretleyen zaman damgasını bulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
```

```
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayStart(InvDate, 1) AS DStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve daystart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	DStart
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00
5/2/2013	07/02/2013 00:00:00
31/3/2013	01/04/2013 00:00:00
19/5/2013	20/05/2013 00:00:00
15/9/2013	16/09/2013 00:00:00
11/12/2013	12/12/2013 00:00:00
2/3/2014	03/03/2014 00:00:00
14/5/2014	15/05/2014 00:00:00
13/6/2014	14/06/2014 00:00:00
7/7/2014	08/07/2014 00:00:00
4/8/2014	05/08/2014 00:00:00

firstworkdate

firstworkdate fonksiyonu, isteğe bağlı olarak listelenen tüm tatilleri dikkate alarak, **end_date** tarihinden önce biten **no_of_workdays** (Pazartesi - Cuma) değerini elde etmek için en son başlangıç tarihini döndürür. **end_date** ve **holiday** geçerli tarihler veya zaman damgaları olmalıdır.

Söz Dizimi:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
end_date	Değerlendirilecek bitiş tarihinin zaman damgası.
no_of_ workdays	Elde edilecek iş günü sayısı.
holiday	İş günlerinden hariç tutulacak tatil dönemleri. Tatil dönemi, virgülle ayrılan bir başlangıç tarihi ve bir bitiş tarihi olarak belirtilir. **Örnek:** `'25/12/2013', '26/12/2013'` Virgüllerle ayrılmış olarak birden fazla tatil dönemi belirtebilirsiniz. **Örnek:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`firstworkdate ('29/12/2014', 9)`	'17/12/2014 döndürür.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	İki günlük bir tatil dönemi de hesaba katıldığından 15/12/2014 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen FirstWorkDate değerlerini gösterir.

Sonuçlar tablosu

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

Bu fonksiyon, sistem saatinden ve Windows saat ayarlarından türetilen geçerli Greenwich Mean Time değerini döndürür.

Söz Dizimi:

```
GMT( )
```

Dönüş verileri türü: dual

Örnek:

```
gmt( )
```

hour

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, saati temsil eden bir tamsayı döndürür.

Söz Dizimi:

```
hour(expression)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`hour( '09:14:36' )`	9 döndürür
`hour( '0.5555' )`	13 sonucunu döndürür (Çünkü 0,5555 = 13:19:55)

inday

Bu fonksiyon, **timestamp** değerinin **base_timestamp** değerini içeren gün içinde olması halinde True döndürür.

Söz Dizimi:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_timestamp** ile karşılaştırmak istediğiniz tarih ve saat.
base_ timestamp	Zaman damgasını değerlendirmek için kullanılan tarih ve saat.
period_no	Gün **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_timestamp** değerini içeren günü gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki günleri; pozitif değerler ise sonraki günleri gösterir.
day_start	Gece yarısı başlamayan günlerle çalışmak istiyorsanız, **day_start** içinde bir günün kesri cinsinden bir kaydırma belirtin; örneğin saat 03:00'ü ifade etmek için 0,125 belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	True döndürür
`inday ('12/01/2006 12:23:00', '13/01/2006 00:00', 0)`	False döndürür
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	False döndürür
`inday ('11/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	True döndürür

Örnek	Sonuç
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0, 0.5)`	False döndürür
`inday ('12/01/2006 11:23:00', '12/01/2006 00:00:00', 0, 0.5)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin base_timestamp ile başlayan gün içinde herhangi bir saate denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InDay(InvTime, '28/03/2012 00:00:00', 0) AS InDayEx
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inday() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvTime	InDayEx
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)

InvTime	InDayEx
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

indaytotime

Bu fonksiyon, **timestamp** öğesi günün **base_timestamp** öğesinin tam milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_timestamp** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

Söz Dizimi:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_timestamp** ile karşılaştırmak istediğiniz tarih ve saat.
base_ timestamp	Zaman damgasını değerlendirmek için kullanılan tarih ve saat.
period_no	Gün **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_timestamp** değerini içeren günü gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki günleri; pozitif değerler ise sonraki günleri gösterir.
day_start	(isteğe bağlı) Gece yarısı başlamayan günlerle çalışmak istiyorsanız, **day_start** içinde bir günün kesri cinsinden bir kaydırma belirtin; örneğin saat 03:00'ü ifade etmek için 0,125 belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 23:59:00', 0)`	True döndürür

Örnek	Sonuç
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	False döndürür
`indaytotime ('11/01/2006 12:23:00', '12/01/2006 23:59:00', -1)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura zaman damgasının base_timestamp ile başlayan gün içinde saat 17:00:00'den öncesine denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InDayToTime(InvTime, '28/03/2012 17:00:00', 0) AS InDayExTT
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve indaytotime() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvTime	InDayExTT
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)

InvTime	InDayExTT
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweek

Bu fonksiyon, **timestamp**'ın **base_date**'i içeren ay haftasında olup olmadığını bulur. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

Söz Dizimi:

```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Ay haftasını değerlendirmek için kullanılan tarih.
period_no	Ay haftası **period_no** ile kaydırılabilir. period_no bir tamsayı olup, burada 0 değeri **base_date** içeren ay haftasını belirtir. **period_no** içindeki negatif değerler önceki ay haftalarını; pozitif değerler ise sonraki ay haftalarını gösterir.
first_week_day	Kaydırma değeri sıfırdan büyük ya da küçük olabilir. Bu değer, belirtilen gün sayısı ve/veya bir günün kesirleri ile yılın başını değiştirir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inlunarweek ('12/01/2013', '14/01/2013', 0)`	True döndürür. Çünkü timestamp değeri (12/01/2013) 08/01/2013 ile 14/01/2013 tarihleri arasındaki haftaya denk gelmektedir.

Örnek	Sonuç
`inlunarweek ('12/01/2013', '07/01/2013', 0)`	False döndürür. Çünkü base_date değeri (07/01/2013) 01/01/2013 ila 07/01/2013 olarak tanımlanan ay haftası içindedir.
`inlunarweek ('12/01/2013', '14/01/2013', -1)`	False döndürür. Çünkü period_no değerinin -1 olarak belirtilmesi haftayı bir önceki haftaya (01/01/2013 ile 07/01/2013 arası) kaydırır.
`inlunarweek ('07/01/2013', '14/01/2013', -1)`	True döndürür. Önceki örnek ile karşılaştırıldığında zaman damgası, geriye doğru kayma hesaba katıldıktan sonraki hafta içindedir.
`inlunarweek ('11/01/2006', '08/01/2006', 0, 3)`	False döndürür. Çünkü first_week_day için değerin 3 olarak belirtilmesi yıl başının 04/01/2013 tarihinden itibaren hesaplanması anlamına gelir. Dolayısıyla, base_date değeri ilk haftaya denk gelir ve timestamp değeri de 11/01/2013 ile 17/01/2013 tarihleri arasındaki haftaya denk gelir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin base_date değerinden dört hafta kaydırılan haftaya denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InLunarWeek(InvDate, '11/01/2013', 4) AS InLWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inlunarweek() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

base_date, 11/01/2013 değeri dört hafta kaydırılıp 5/02/2013 ile 11/02/2013 tarihleri arasındaki haftaya denk geldiğinden fonksiyon, InvDate5/2/2013 değeri için True sonucunu döndürür.

Sonuçlar tablosu

InvDate	InLWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweektodate

Bu fonksiyon, **timestamp**'ın ay haftası ile **base_date**'in son milisaniyesi arasında yer alıp almadığını bulur. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

Söz Dizimi:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Ay haftasını değerlendirmek için kullanılan tarih.
period_no	Ay haftası **period_no** ile kaydırılabilir. period_no bir tamsayı olup, burada 0 değeri **base_date** içeren ay haftasını belirtir. **period_no** içindeki negatif değerler önceki ay haftalarını; pozitif değerler ise sonraki ay haftalarını gösterir.

Bağımsız Değişken	Açıklama
first_week_day	Kaydırma değeri sıfırdan büyük ya da küçük olabilir. Bu değer, belirtilen gün sayısı ve/veya bir günün kesirleri ile yılın başını değiştirir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inlunarweektodate ('12/01/2013', '13/01/2013', 0)`	True döndürür. Çünkü timestamp değeri (12/01/2013) haftanın 08/01/2013 ile 13/01/2013 tarihleri arasındaki bölümüne denk gelmektedir.
`inlunarweektodate ('12/01/2013', '11/01/2013', 0)`	False döndürür. Çünkü timestamp değeri base_date değerinden sonraki bir zamandır (iki tarih de 12/01/2012 tarihinden önceki aynı ay haftasında olsa bile).
`inlunarweektodate ('12/01/2006', '05/01/2006', 1)`	True döndürür. period_no için 1 değerinin belirtilmesi base_date değerini bir hafta ileri kaydırır ve böylece timestamp değeri ay haftası bölümüne denk gelir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin base_date değerinden dört hafta kaydırılan hafta bölümüne denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InLunarWeekToDate(InvDate, '07/01/2013', 4) AS InLWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inlunarweek() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

base_date, 11/01/2013 değeri dört hafta kaydırılıp 5/02/2013 ile 07/02/2013 tarihleri arasındaki haftaya denk geldiğinden fonksiyon, InvDate5/2/2013 değeri için True sonucunu döndürür.

Sonuçlar tablosu

InvDate	InLWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonth

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren ay içinde olması halinde True döndürür.

Söz Dizimi:

```
InMonth (timestamp, base_date, period_no[, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Ayı değerlendirmek için kullanılan tarih.

Bağımsız Değişken	Açıklama
period_no	Ay **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren ayı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki ayları; pozitif değerler ise sonraki ayları gösterir.
first_month_of_year	**first_month_of_year** parametresi devre dışı durumdadır ve gelecekte kullanılmak üzere ayrılmıştır.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inmonth ('25/01/2013', '01/01/2013', 0 )`	True döndürür
`inmonth('25/01/2013', '01/04/2013', 0)`	False döndürür
`inmonth ('25/01/2013', '01/01/2013', -1)`	False döndürür
`inmonth ('25/12/2012', '01/01/2013', -1)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, period_no değerinin 4 olarak belirtilmesiyle, bir fatura tarihinin base_date içinde verilen aydan sonraki dördüncü ay içinde herhangi bir zamana denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonth(InvDate, '31/01/2013', 4) AS InMthPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inmonth() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	InMthPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	-1 (True)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonths

Bu fonksiyon, bir zaman damgasının taban tarih ile aynı ay, iki aylık dönem, çeyrek, tersiyel veya yarım yıl içine denk gelip gelmediğini bulur. Zaman damgasının önceki veya sonraki bir zaman dönemine denk gelip gelmediğini bulmak da mümkündür.

Söz Dizimi:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n_months	Dönemi tanımlayan ayların sayısı. Şunlardan biri olması gereken bir tamsayı veya bir tamsayıya çözümlenen bir ifade: 1 (inmonth() fonksiyonuna eşdeğer), 2 (iki aylık), 3 (inquarter() fonksiyonuna eşdeğer), 4 (dört aylık), or 6 (altı aylık).
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Dönemi değerlendirmek için kullanılan tarih.

Bağımsız Değişken	Açıklama
period_no	Dönem **period_no** ile kaydırılabilir. Bu değer bir tamsayı ya da tamsayıya çözümlenen bir ifadedir ve burada 0 değeri **base_date** içeren dönemi belirtir. **period_no** içindeki negatif değerler önceki dönemleri; pozitif değerler ise sonraki dönemleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`inmonths(4, '25/01/2013', '25/04/2013', 0)`	True döndürür. Çünkü timestamp değeri (25/01/2013) 01/01/2013 ile 30/04/2013 tarihleri arasındaki dört aylık dönem içinde yer almaktadır ve base_date değeri 25/04/2013 de bu dönem içindedir.
`inmonths(4, '25/05/2013', '25/04/2013', 0)`	False döndürür. Çünkü 25/05/2013 yukarıdaki örnekte verilen aynı dönemin dışındadır.
`inmonths(4, '25/11/2012', '01/02/2013', -1 )`	True döndürür. Çünkü period_no için -1 değeri, arama dönemini dört aylık bir dönem kadar (n-months değeri) geriye kaydırır ve bu da arama dönemini 01/09/2012 ile 31/12/2012. tarihleri arasına getirir.
`inmonths( 4, '25/05/2006', '01/03/2006', 0, 3)`	True döndürür. Çünkü first_month_of_year değeri 3 olarak ayarlanmıştır ve bu da arama dönemini 01/03/2006 ile 30/07/2006 tarihleri arasına getirir (01/01/2006 ila 30/04/2006 aralığı yerine).

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki fatura tarihinin, iki aylık bir dönem kadar kaydırılan (period_no değerinin 1 olarak belirtilmesiyle) base_date değerinin içinde bulunduğu iki aylık döneme denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonths(2, InvDate, '11/02/2013', 1) AS InMthsPlus1
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve InMonths() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Arama dönemi 01/03/2013 ile 30/04/2013 tarihleri arasındadır; çünkü base_date değeri fonksiyondaki değerden (11/02/2013) iki ay ileri kaydırılmaktadır.

Sonuçlar tablosu

InvDate	InMthsPlus1
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthstodate

Bu fonksiyon, bir zaman damgasının ayın, iki ayın, üç ayın, dört ayın veya altı ayın parçası ve **base_date**'in son milisaniyesi içinde olup olmadığını bulur. Zaman damgasının önceki veya sonraki bir zaman dönemine denk gelip gelmediğini bulmak da mümkündür.

Söz Dizimi:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n_months	Dönemi tanımlayan ayların sayısı. Şunlardan biri olması gereken bir tamsayı veya bir tamsayıya çözümlenen bir ifade: 1 (inmonth() fonksiyonuna eşdeğer), 2 (iki aylık), 3 (inquarter() fonksiyonuna eşdeğer), 4 (dört aylık), or 6 (altı aylık).
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Dönemi değerlendirmek için kullanılan tarih.
period_no	Dönem **period_no** ile kaydırılabilir. Bu değer bir tamsayı ya da tamsayıya çözümlenen bir ifadedir ve burada 0 değeri **base_date** içeren dönemi belirtir. **period_no** içindeki negatif değerler önceki dönemleri; pozitif değerler ise sonraki dönemleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`inmonthstodate(4, '25/01/2013', '25/04/2013', 0)`	True döndürür. Çünkü timestamp değeri (25/01/2013) 01/01/2013 tarihinden itibaren 25/04/2013 sonuna kadarki dört aylık dönem içinde yer almaktadır ve base_date değeri 25/04/2013 de bu dönem içindedir.
`inmonthstodate(4, '26/04/2013', '25/04/2006', 0)`	False döndürür. Çünkü 26/04/2013 yukarıdaki örnekte verilen aynı dönemin dışındadır.

Örnek	Sonuç
`inmonthstodate(4, '25/09/2005', '01/02/2006', -1)`	True döndürür. Çünkü period_no için -1 değeri, arama dönemini dört aylık bir dönem kadar (n-months değeri) geriye kaydırır ve bu da arama dönemini 01/09/2005 ile 01/02/2006. tarihleri arasına getirir.
`inmonthstodate(4, '25/04/2006', '01/06/2006', 0, 3)`	True döndürür. Çünkü first_month_of_year değeri 3 olarak ayarlanmıştır ve bu da arama dönemini 01/03/2006 ile 01/06/2006 tarihleri arasına getirir (01/05/2006 ila 01/06/2006 aralığı yerine).

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki fatura tarihinin, ikişer aylık dört dönem kadar kaydırılan (period_no değerinin 4 olarak belirtilmesiyle) base_date değerine kadarki (bu tarih dahil) iki aylık dönem bölümüne denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonthsToDate(2, InvDate, '15/02/2013', 4) AS InMths2DPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve InMonths() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Arama dönemi 01/09/2013 ile 15/10/2013 tarihleri arasındadır; çünkü base_date değeri fonksiyondaki değerden (15/02/2013) sekiz ay ileri kaydırılmaktadır.

Sonuçlar tablosu

InvDate	InMths2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	-1 (True)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthtodate

date, ayın **basedate** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **basedate** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

Söz Dizimi:

```
InMonthToDate (timestamp, base_date, period_no)
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Ayı değerlendirmek için kullanılan tarih.
period_no	Ay **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren ayı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki ayları; pozitif değerler ise sonraki ayları gösterir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inmonthtodate ('25/01/2013', '25/01/2013', 0)`	True döndürür
`inmonthtodate ('25/01/2013', '24/01/2013', 0)`	False döndürür
`inmonthtodate ('25/01/2013', '28/02/2013', -1)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, period_no değerinin 4 olarak belirtilmesiyle, bir fatura tarihinin base_date içinde verilen aydan sonraki dördüncü aya (ancak base_date içinde belirtilen günün bitiminden öncesine) denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonthToDate(InvDate, '31/01/2013', 4) AS InMthPlus42D
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inmonthtodate() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	InMthPlus42D
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)

InvDate	InMthPlus42D
31/3/2013	0 (False)
19/5/2013	-1 (True)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquarter

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren çeyrek içinde olması halinde True döndürür.

Söz Dizimi:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Çeyreği değerlendirmek için kullanılan tarih.
period_no	Çeyrek **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren çeyreği gösterdiği bir tam sayıdır. **period_no** içindeki negatif değerler önceki çeyrekleri; pozitif değerler ise sonraki çeyrekleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inquarter ('25/01/2013', '01/01/2013', 0)`	True döndürür

Örnek	Sonuç
`inquarter ('25/01/2013', '01/04/2013', 0)`	False döndürür
`inquarter ('25/01/2013', '01/01/2013', -1)`	False döndürür
`inquarter ('25/12/2012', '01/01/2013', -1)`	True döndürür
`inquarter ('25/01/2013', '01/03/2013', 0, 3)`	False döndürür
`inquarter ('25/03/2013', '01/03/2013', 0, 3)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin, first_month_of_year değerinin 4 olarak ayarlanması ve base_date olarak 31/01/2013 kullanılması ile belirtilen mali yılın dördüncü çeyreğine denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InQuarter(InvDate, '31/01/2013', 0, 4) AS Qtr4FinYr1213
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inquarter() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	Qtr4Fin1213
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)

InvDate	Qtr4Fin1213
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquartertodate

Bu fonksiyon, **timestamp** öğesi çeyreğin **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

Söz Dizimi:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Çeyreği değerlendirmek için kullanılan tarih.
period_no	Çeyrek **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren çeyreği gösterdiği bir tam sayıdır. **period_no** içindeki negatif değerler önceki çeyrekleri; pozitif değerler ise sonraki çeyrekleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inquartertodate ('25/01/2013', '25/01/2013', 0)`	True döndürür
`inquartertodate ( 25/01/2013', '24/01/2013', 0)`	False döndürür
`inquartertodate ('25/01/2012', '01/02/2013', -1)`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin, first_month_of_year değerinin 4 olarak ayarlanması ile belirtilen mali yıla ve de dördüncü çeyrekte 28/02/2013 gün bitiminden öncesine denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InQuarterToDate(InvDate, '28/02/2013', 0, 4) AS Qtr42Date
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inquartertodate() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	Qtr42Date
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)

InvDate	Qtr42Date
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inweek

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren hafta içinde olması halinde True döndürür.

Söz Dizimi:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Haftayı değerlendirmek için kullanılan tarih.
period_no	Hafta **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren haftayı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki haftaları; pozitif değerler ise sonraki haftaları gösterir.
first_week_day	Varsayılan olarak, haftanın ilk günü Pazar ile Pazartesi arasındaki gece yarısından başlayan Pazartesi'dir. Haftanın başka bir gün başlamasını belirtmek için **first_week_day** içinde bir kaydırma belirtin. Bu, günleri ve/veya bir günün kesirlerini belirten bir bütün sayı olarak verilebilir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inweek ('12/01/2006', '14/01/2006', 0)`	True döndürür
`inweek ('12/01/2006', '20/01/2006', 0 )`	False döndürür
`inweek ('12/01/2006', '14/01/2006', -1 )`	False döndürür
`inweek ('07/01/2006', '14/01/2006', -1)`	True döndürür
`inweek ('12/01/2006', '09/01/2006', 0, 3)`	False döndürür Bunun sebebi first_week_day değerinin, 3 (Perşembe) olarak belirtilmesi ve bunun da 12/01/2006 tarihini, 09/01/2006 tarihini içeren haftayı takip eden haftanın ilk günü yapmasıdır.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, period_no değerinin 4 olarak belirtilmesiyle, bir fatura tarihinin base_date içinde verilen haftadan sonraki dördüncü hafta içinde herhangi bir zamana denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InWeek(InvDate, '11/01/2013', 4) AS InWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inweek() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

InvDate5/2/2013, base_date tarihinden dört hafta sonraki haftanın içine denk gelir: 11/1/2013.

Sonuçlar tablosu

InvDate	InWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inweektodate

Bu fonksiyon, **timestamp**, haftanın **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

Söz Dizimi:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Haftayı değerlendirmek için kullanılan tarih.

Bağımsız Değişken	Açıklama
period_no	Hafta **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren haftayı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki haftaları; pozitif değerler ise sonraki haftaları gösterir.
first_week_day	Varsayılan olarak, haftanın ilk günü Pazar ile Pazartesi arasındaki gece yarısından başlayan Pazartesi'dir. Haftanın başka bir gün başlamasını belirtmek için **first_week_day** içinde bir kaydırma belirtin. Bu, günleri ve/veya bir günün kesirlerini belirten bir bütün sayı olarak verilebilir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inweektodate ('12/01/2006', '12/01/2006', 0)`	True döndürür
`inweektodate ('12/01/2006', '11/01/2006', 0)`	False döndürür
`inweektodate ('12/01/2006', '18/01/2006', -1)`	False döndürür period_no, -1 olarak belirtildiğinden, timestamp değerinin hesaplanmasında temel alınan geçerli tarih 11/01/2006 olur.
`inweektodate ( '11/01/2006', '12/01/2006', 0, 3 )`	False döndürür Bunun sebebi first_week_day değerinin, 3 (Perşembe) olarak belirtilmesi ve bunun da 12/01/2006 tarihini, 12/01/2006 tarihini içeren haftayı takip eden haftanın ilk günü yapmasıdır.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, period_no değerinin 4 olarak belirtilmesiyle, bir fatura tarihinin base_date içinde verilen haftadan sonraki dördüncü haftaya (ancak base_date değerinden öncesine) denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
```

```
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InWeekToDate(InvDate, '11/01/2013', 4) AS InWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inweek() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	InWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyear

Bu fonksiyon, **timestamp** değerinin **base_date** değerini içeren yıl içinde olması halinde True döndürür.

Söz Dizimi:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Yılı değerlendirmek için kullanılan tarih.
period_no	Yıl **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren yılı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki yılları; pozitif değerler ise sonraki yılları gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`inyear ('25/01/2013', '01/01/2013', 0 )`	True döndürür
`inyear ('25/01/2012', '01/01/2013', 0)`	False döndürür
`inyear ('25/01/2013', '01/01/2013', -1)`	False döndürür
`inyear ('25/01/2012', '01/01/2013', -1 )`	True döndürür
`inyear ('25/01/2013', '01/01/2013', 0, 3)`	True döndürür base_date ve first_month_of_year değeri, timestamp değerinin 01/03/2012 ile 28/02/2013 tarihleri arasına denk gelmesi gerektiğini belirtir
`inyear ('25/03/2013', '01/07/2013', 0, 3 )`	True döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin, first_month_of_year değerinin 4 olarak ayarlanması ve base_date olarak 1/4/2012 ile 31/03/2013 arasının kullanılması ile belirtilen mali yıla denk gelip gelmediğini kontrol eder.

```
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
];
```

InvDate değerinin 1/04/2012 ila 31/03/2013 mali yılında olup olmadığını test edin:

```
InvoiceData:
LOAD *,
InYear(InvDate, '31/01/2013', 0, 4) AS FinYr1213
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inyear() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	FinYr1213
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	-1 (True)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyeartodate

Bu fonksiyon, **timestamp**, yılın **base_date** öğesinin son milisaniyesine kadar ve bu milisaniye de dahil olmak üzere **base_date** öğesini içeren bölümünde bulunuyorsa, True sonucunu döndürür.

Söz Dizimi:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	**base_date** ile karşılaştırmak istediğiniz tarih.
base_date	Yılı değerlendirmek için kullanılan tarih.
period_no	Yıl **period_no** ile kaydırılabilir. **period_no**, 0 değerinin **base_date** değerini içeren yılı gösterdiği bir tamsayıdır. **period_no** içindeki negatif değerler önceki yılları; pozitif değerler ise sonraki yılları gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`inyeartodate ('2013/01/25', '2013/02/01', 0)`	True döndürür
`inyeartodate ('2012/01/25', '2013/01/01', 0)`	False döndürür
`inyeartodate ('2012/01/25', '2013/02/01', -1)`	True döndürür
`inyeartodate ('2012/11/25', '2013/01/31', 0, 4)`	True döndürür timestamp değeri, dördüncü ayda başlayan mali yılın içine ve base_date değerinden öncesine denk gelir.

Örnek	Sonuç
`inyeartodate ( '2013/3/31', '2013/01/31', 0, 4 )`	False döndürür Önceki örnek ile karşılaştırıldığında timestamp değeri halen ilgili mali yılın içindedir; ancak base_date değerinden sonra olduğundan yıl bölümünün dışında kalır.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, bir fatura tarihinin, first_month_of_year değerinin 4 olarak ayarlanması ile belirtilen mali yıla ve de bu yılın 31/01/2013 gün bitiminden önceki bölümüne denk gelip gelmediğini kontrol eder.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InYearToDate(InvDate, '31/01/2013', 0, 4) AS FinYr2Date
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve inyeartodate() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	FinYr2Date
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)

InvDate	FinYr2Date
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

lastworkdate

lastworkdate fonksiyonu, isteğe bağlı **holiday** varsa bunları da dikkate alarak, **start_date** ile başlanması halinde **no_of_workdays** (Pazartesi-Cuma) elde edilmesi için gerekli en erken bitiş tarihini döndürür. **start_date** ve **holiday** geçerli tarihler veya zaman damgaları olmalıdır.

Söz Dizimi:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
start_date	Değerlendirilecek başlangıç tarihi.
no_of_ workdays	Elde edilecek iş günü sayısı.
holiday	İş günlerinden hariç tutulacak tatil dönemleri. Tatil dönemi, virgülle ayrılan bir başlangıç tarihi ve bir bitiş tarihi olarak belirtilir. **Örnek:** `'25/12/2013', '26/12/2013'` Virgüllerle ayrılmış olarak birden fazla tatil dönemi belirtebilirsiniz. **Örnek:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`lastworkdate ('19/12/2014', 9)`	'31/12/2014' döndürür
`lastworkdate ('19/12/2014', 9, '2014-12-25', '2014-12-26')`	İki günlük bir tatil dönemi de hesaba katıldığından 02/01/2015 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
ProjectTable:
LOAD *, recno() as InvID, INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
LastWorkDate(StartDate,120) As EndDate
Resident ProjectTable;
Drop table ProjectTable;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen LastWorkDate değerlerini gösterir.

Sonuçlar tablosu

InvID	StartDate	EndDate
1	28/03/2014	11/09/2014
2	10/12/2014	26/05/2015
3	5/2/2015	27/07/2015
4	31/3/2015	14/09/2015
5	19/5/2015	02/11/2015
6	15/9/2015	29/02/2016

localtime

Bu fonksiyon, belirtilen bir saat dilimi için sistem saatinden geçerli zamanın zaman damgasını döndürür.

Söz Dizimi:

```
LocalTime([timezone [, ignoreDST ]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timezone	**timezone**, **Date and Time** için **Windows Control Panel** içindeki **Time Zone** altında listelenen coğrafi konumlardan herhangi birini içeren bir dize olarak veya 'GMT+hh:mm' biçiminde bir dize olarak belirtilir. Herhangi bir saat dilimi belirtilmezse yerel zaman döndürülür.
ignoreDST	**ignoreDST** öğesi -1 (True) ise günışığından yararlanma saati göz ardı edilir.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde fonksiyonun 2014-10-22 12:54:47 yerel saatinde çağrıldığı ve yerel saat diliminin GMT+01:00 olduğu esas alınmaktadır.

Kod örnekleri

Örnek	Sonuç
`localtime ()`	Yerel saati (2014-10-22 12:54:47) döndürür.
`localtime ('London')`	Londra'daki yerel saati (2014-10-22 11:54:47) döndürür.
`localtime ('GMT+02:00')`	GMT+02:00 zaman dilimindeki yerel saati (2014-10-22 13:54:47) döndürür.
`localtime ('Paris','-1')`	Günışığından yararlanma saatini yok sayarak, Paris'teki yerel saati (2014-10-22 11:54:47) döndürür.

lunarweekend

Bu fonksiyon, **date** içeren ay haftasının son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

Söz Dizimi:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no**, tamsayıya çözümlenen bir tamsayı veya ifade olup, burada 0 değeri **date** içeren ay haftasını belirtir. **period_no** içindeki negatif değerler önceki ay haftalarını; pozitif değerler ise sonraki ay haftalarını gösterir.
first_week_day	Kaydırma değeri sıfırdan büyük ya da küçük olabilir. Bu değer, belirtilen gün sayısı ve/veya bir günün kesirleri ile yılın başını değiştirir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`lunarweekend('12/01/2013')`	14/01/2013 23:59:59 döndürür.
`lunarweekend('12/01/2013', -1)`	7/01/2013 23:59:59 döndürür.
`lunarweekend('12/01/2013', 0, 1)`	15/01/2013 23:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihi için ay haftasının son gününü bulur. Burada date öğesi, period_no değerinin 1 olarak belirtilmesiyle bir hafta kaydırılmaktadır.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
```

```
4/8/2014
];

InvoiceData:
LOAD *,
LunarWeekEnd(InvDate, 1) AS LWkEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve lunarweekend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	LWkEnd
28/03/2012	07/04/2012
10/12/2012	22/12/2012
5/2/2013	18/02/2013
31/3/2013	08/04/2013
19/5/2013	27/05/2013
15/9/2013	23/09/2013
11/12/2013	23/12/2013
2/3/2014	11/03/2014
14/5/2014	27/05/2014
13/6/2014	24/06/2014
7/7/2014	15/07/2014
4/8/2014	12/08/2014

lunarweekname

Bu fonksiyon, **date** içeren ay haftasının ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen yıl ve ay haftası numarasını gösteren bir görüntü değeri döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

Söz Dizimi:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no**, tamsayıya çözümlenen bir tamsayı veya ifade olup, burada 0 değeri **date** içeren ay haftasını belirtir. **period_no** içindeki negatif değerler önceki ay haftalarını; pozitif değerler ise sonraki ay haftalarını gösterir.
first_week_day	Kaydırma değeri sıfırdan büyük ya da küçük olabilir. Bu değer, belirtilen gün sayısı ve/veya bir günün kesirleri ile yılın başını değiştirir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`lunarweekname('12/01/2013')`	2006/02 döndürür.
`lunarweekname('12/01/2013', -1)`	2006/01 döndürür.
`lunarweekname('12/01/2013', 0, 1)`	2006/02 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnekte, tablodaki her bir fatura tarihi için, haftanın içinde yer aldığı yıldan ve period_no değerinin 1 olarak belirtilmesiyle bir hafta kaydırılmış ilişkili ay haftası sayısından ay haftasının adı oluşturulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

```
InvoiceData:
LOAD *,
LunarWeekName(InvDate, 1) AS LWkName
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve lunarweekname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	LWkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/28
4/8/2014	2014/32

lunarweekstart

Bu fonksiyon, **date** içeren ay haftasının ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Qlik Sense içindeki ay haftaları, haftanın ilk günü 1 Ocak sayılarak tanımlanır.

Söz Dizimi:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no**, tamsayıya çözümlenen bir tamsayı veya ifade olup, burada 0 değeri **date** içeren ay haftasını belirtir. **period_no** içindeki negatif değerler önceki ay haftalarını; pozitif değerler ise sonraki ay haftalarını gösterir.
first_week_day	Kaydırma değeri sıfırdan büyük ya da küçük olabilir. Bu değer, belirtilen gün sayısı ve/veya bir günün kesirleri ile yılın başını değiştirir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`lunarweekstart ('12/01/2013')`	08/01/2013 döndürür.
`lunarweekstart ('12/01/2013', -1)`	01/01/2013 döndürür.
`lunarweekstart ('12/01/2013', 0, 1 )`	09/01/2013 döndürür. Çünkü first_week_day değerinin 1 olarak ayarlanmasıyla belirtilen kayma, yılın başlangıcının 02/01/2013 olarak değişmesi anlamına gelir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihi için ay haftasının ilk gününü bulur. Burada date öğesi, period_no değerinin 1 olarak belirtilmesiyle bir hafta kaydırılmaktadır.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
```

```
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
LunarWeekStart(InvDate, 1) AS LWkStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve lunarweekstart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	LWkStart
28/03/2012	01/04/2012
10/12/2012	16/12/2012
5/2/2013	12/02/2013
31/3/2013	02/04/2013
19/5/2013	21/05/2013
15/9/2013	17/09/2013
11/12/2013	17/12/2013
2/3/2014	05/03/2014
14/5/2014	21/05/2014
13/6/2014	18/06/2014
7/7/2014	09/07/2014
4/8/2014	06/08/2014

makedate

Bu fonksiyon **YYYY** yılı, **MM** ayı ve **DD** gününden hesaplanan bir tarih döndürür.

Söz Dizimi:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
YYYY	Tamsayı olarak yıl.
MM	Tamsayı olarak ay. Ay belirtilmezse 1 (Ocak) olduğu varsayılır.
DD	Tamsayı olarak gün. Gün belirtilmezse 1. (birinci) gün olduğu varsayılır.

Örnek: Grafik ifadesi

Grafik ifadesi örnekleri

Örnek	Sonuç
`makedate(2012)`	şunu döndürür: 2012-01-01
`makedate(12)`	şunu döndürür: 0012-01-01
`makedate(2012,12)`	şunu döndürür: 2012-12-01
`makedate(2012,2,14)`	şunu döndürür: 2012-02-14

Örnek: Yükleme kodu
makedate, farklı alanlardaki tarih verilerini tek bir yeni tarih alanında birleştirmek için yükleme kodunda kullanılabilir. Aşağıdaki örnekte, transaction_year, transaction_month, ve transaction_day alanlarındaki yıl, ay ve gün verileri, İşlem Tarihi adlı yeni bir alanda birleştirilir.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

Yükleme kodu

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
Load
*,
MakeDate(transaction_year, transaction_month, transaction_day) as "Transaction Date",
;

Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
```

```
transaction_quantity, discount, customer_id, size, color_code
3750, 2018, 08, 30, 12423.56, 23, 0,2038593, L, Red
3751, 2018, 09, 07, 5356.31, 6, 0.1, 203521, m, orange
3752, 2018, 09, 16, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018, 09, 22, 1251, 7, 0, 3036491, l, Black
3754, 2018, 09, 22, 21484.21, 1356, 75, 049681, xs, Red
3756, 2018, 09, 22, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 2018, 09, 23, 3177.4, 21, .14, 203521, XL, Black
];
```

Sonuçlar

Qlik Sense table showing results of the makedate function being used in the load script.

transaction_id	Transaction Date
3750	30/08/2018
3751	07/09/2018
3752	16/09/2018
3753	22/09/2018
3754	22/09/2018
3756	22/09/2018
3757	23/09/2018

maketime

Bu fonksiyon **hh** saati, **mm** dakikası ve **ss** saniyesinden hesaplanan bir saat döndürür.

Söz Dizimi:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
hh	Tamsayı olarak saat.
mm	Tamsayı olarak dakika. Dakika belirtilmezse 00 olduğu varsayılır.
ss	Tamsayı olarak saniye. Saniye belirtilmezse 00 olduğu varsayılır.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`maketime( 22 )`	şunu döndürür: 22:00:00
`maketime( 22, 17 )`	şunu döndürür: 22:17:00
`maketime( 22, 17, 52 )`	şunu döndürür: 22:17:52

makeweekdate

Bu fonksiyon **YYYY** yılı, **WW** haftası ve **D** haftanın gününden hesaplanan bir tarih döndürür.

Söz Dizimi:

```
MakeWeekDate(YYYY [ , WW [ , D ] ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
YYYY	Tamsayı olarak yıl.
WW	Tamsayı olarak hafta.
D	Tamsayı olarak haftanın günü. Haftanın günü belirtilmezse 0 (Pazartesi) olduğu varsayılır.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`makeweekdate(2014,6,6)`	şunu döndürür: 2014-02-09
`makeweekdate(2014,6,1)`	şunu döndürür: 2014-02-04
`makeweekdate(2014,6)`	2014-02-03 döndürür (haftanın 0. günü olduğu varsayılır)

minute

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, dakikayı temsil eden bir tamsayı döndürür.

Söz Dizimi:

```
minute(expression)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`minute ( '09:14:36' )`	14 döndürür
`minute ( '0.5555' )`	19 sonucunu döndürür (Çünkü 0,5555 = 13:19:55)

month

Bu fonksiyon, ikili değer döndürür: **MonthNames** ortam değişkeninde tanımlandığı şekliyle ay adı ve 1-12 arasında bir tamsayı. Ay, standart sayı yorumlamasına göre ifadenin tarih yorumlamasından hesaplanır.

Fonksiyon, belirli bir tarih için `MonthName` sistem değişkeninin formatında ayın adını döndürür. Sıklıkla bir Ana Takvim'de bir boyut olarak gün alanı oluşturmak için kullanılır.

Söz Dizimi:

```
month(expression)
```

Dönüş verileri türü: tamsayı

Fonksiyon örnekleri

Örnek	Sonuç
`month( '2012-10-12' )`	Eki sonucunu döndürür
`month( '35648' )`	35648 = 1997-08-06 olduğundan Ağu sonucunu döndürür

monthend

Bu fonksiyon, **date** içeren ayın son gününün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
MonthEnd(date[, period_no])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, 0 olur ya da atlanırsa **date** içeren ayı belirtir. **period_no** içindeki negatif değerler önceki ayları; pozitif değerler ise sonraki ayları gösterir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`monthend('19/02/2012')`	29/02/2012 23:59:59 döndürür.
`monthend('19/02/2001', -1)`	31/01/2001 23:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin ayı içinde son günü bulur. Burada taban tarih, *period_no* değerinin 4 olarak belirtilmesiyle dört hafta kaydırılmaktadır.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthEnd(InvDate, 4) AS MthEnd
```

```
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve monthend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	MthEnd
28/03/2012	31/07/2012
10/12/2012	30/04/2013
5/2/2013	30/06/2013
31/3/2013	31/07/2013
19/5/2013	30/09/2013
15/9/2013	31/01//2014
11/12/2013	30/04//2014
2/3/2014	31/07//2014
14/5/2014	30/09/2014
13/6/2014	31/10/2014
7/7/2014	30/11/2014
4/8/2014	31/12/2014

monthname

Bu fonksiyon, ayı (**MonthNames** kod değişkenine göre biçimlendirilmiş) ve yılı, ayın ilk gününün ilk milisaniyesine sahip zaman damgasına karşılık gelen bir temel sayısal değerle gösteren bir görüntü değeri döndürür.

Söz Dizimi:

```
MonthName(date[, period_no])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, 0 olur ya da atlanırsa **date** içeren ayı belirtir. **period_no** içindeki negatif değerler önceki ayları; pozitif değerler ise sonraki ayları gösterir.

Örnek: Grafik ifadesi
Bu örnekte, veri yükleme kodunuzun en üstündeki **SET DateFormat** deyiminde belirtilen **DD/MM/YYYY** tarih biçimi kullanılır. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin. **SET Monthnames** deyimi Jan;Feb;Mar vb. olarak ayarlanır.

Grafik ifadesi örnekleri

Örnek	Sonuç
`monthname('19/10/2013')`	Oct 2013 döndürür
`monthname('19/10/2013', -1)`	Sep 2013 döndürür

Örnek: Yükleme kodu
Bu örnekte, tablodaki her bir fatura tarihi için, base_date değerinden dört ay kaydırılmış ay adından ve yıldan ay adı oluşturulur.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

Yükleme kodu

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthName(InvDate, 4) AS MthName
Resident TempTable;
Drop table TempTable;
```

Sonuçlar

Sonuçta ortaya çıkan tabloda orijinal tarihler ve monthname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

InvDate	MthName
28/03/2012	Jul 2012
10/12/2012	Apr 2013
5/2/2013	Jun 2013
31/3/2013	Jul 2013
19/5/2013	Sep 2013
15/9/2013	Jan 2014
11/12/2013	Apr 2014
2/3/2014	Jul 2014
14/5/2014	Sep 2014
13/6/2014	Oct 2014
7/7/2014	Nov 2014
4/8/2014	Dec 2014

Örnek: Yükleme kodu
Bu örnekte, tablodaki her transaction_date için bir Returnable_Until değeri oluşturulur. Returnable_Until değeri, transaction_date ayı bir ay sonrasına kaydırılarak hesaplanır.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

Yükleme kodu

```
SET DateFormat='YYYYMMDD';
SET TimestampFormat='YYYYMMDD h:mm:ss[.fff] TT';
SET FirstMonthOfYear=1;
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
*,
MonthName(Date#(transaction_date,'YYYYMMDD'), 1) as Returnable_Until,
```

```
;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Sonuçlar

Qlik Sense table showing results of the monthname function being used in the load script.

transaction_id	transaction_date	Returnable_Until
3750	20180830	Sep 2018
3751	20180907	Oct 2018
3752	20180916	Oct 2018
3753	20180922	Oct 2018
3754	20180922	Oct 2018
3756	20180922	Oct 2018
3757	20180923	Oct 2018

monthsend

Bu fonksiyon bir taban tarih içeren ayın, iki aylık dönemin, çeyreğin, tersiyelin veya yarım yılın son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Önceki ve sonraki bir zaman dönemi için zaman damgasını bulmak da mümkündür.

Söz Dizimi:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n_months	Dönemi tanımlayan ayların sayısı. Şunlardan biri olması gereken bir tamsayı veya bir tamsayıya çözümlenen bir ifade: 1 (inmonth() fonksiyonuna eşdeğer), 2 (iki aylık), 3 (inquarter() fonksiyonuna eşdeğer), 4 (dört aylık), or 6 (altı aylık).
date	Değerlendirilecek tarih.
period_no	Dönem **period_no** ile kaydırılabilir. Bu değer bir tamsayı ya da tamsayıya çözümlenen bir ifadedir ve burada 0 değeri **base_date** içeren dönemi belirtir. **period_no** içindeki negatif değerler önceki dönemleri; pozitif değerler ise sonraki dönemleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`monthsend(4, '19/07/2013')`	31/08/2013 döndürür.
`monthsend(4, '19/10/2013', -1)`	31/08/2013 döndürür.
`monthsend(4, '19/10/2013', 0, 2)`	31/01/2014 döndürür. Çünkü yılın başlangıcı 2. ay olur.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, her bir fatura tarihi için iki aylık dönemin son gününün bitişini bulur (iki aylık bir dönem kadar ileri kaydırılmış olarak).

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
```

```
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsEnd(2, InvDate, 1) AS BiMthsEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve MonthsEnd() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	BiMthsEnd
28/03/2012	30/06/2012
10/12/2012	28/02/2013
5/2/2013	30/04/2013
31/3/2013	30/04/2013
19/5/2013	31/08/2013
15/9/2013	31/12/2013
11/12/2013	28/02/2014
2/3/2014	30/06/2014
14/5/2014	31/08/2014
13/6/2014	31/08/2014
7/7/2014	31/10/2014
4/8/2014	31/10/2014

monthsname

Bu fonksiyon, dönemin ay aralığının (**MonthNames** kod değişkenine göre biçimlendirilmiş) yanı sıra yılı temsil eden bir görüntü değeri döndürür. Temel sayısal değer; ayın, iki ayın, üç ayın, dört ayın veya altı ayın temel tarih içeren ilk milisaniyesinin zaman damgasına karşılık gelir.

Söz Dizimi:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n_months	Dönemi tanımlayan ayların sayısı. Şunlardan biri olması gereken bir tamsayı veya bir tamsayıya çözümlenen bir ifade: 1 (inmonth() fonksiyonuna eşdeğer), 2 (iki aylık), 3 (inquarter() fonksiyonuna eşdeğer), 4 (dört aylık), or 6 (altı aylık).
date	Değerlendirilecek tarih.
period_no	Dönem **period_no** ile kaydırılabilir. Bu değer bir tamsayı ya da tamsayıya çözümlenen bir ifadedir ve burada 0 değeri **base_date** içeren dönemi belirtir. **period_no** içindeki negatif değerler önceki dönemleri; pozitif değerler ise sonraki dönemleri gösterir.
first_ month_of_ year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`monthsname(4, '19/10/2013')`	'Sep-Dec 2013 döndürür. Çünkü bu ve diğer örneklerde **SET Monthnames** deyimi Jan;Feb;Mar vs. olarak ayarlanmıştır.
`monthsname(4, '19/10/2013', -1)`	'May-Aug 2013 döndürür.
`monthsname(4, '19/10/2013', 0, 2)`	Oct-Jan 2014 döndürür. Çünkü yılın 2. ayda başladığı belirtilmektedir. Bu nedenle, dört aylık dönem takip eden yılın birinci ayında sona erer.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnekte, tablodaki her bir fatura tarihi için, iki aylık dönemdeki ay aralığından ve yıldan ay adı oluşturulur. Aralık 4x2 ay kadar kaydırılmıştır (period_no değerinin 4 olarak belirtilmesiyle).

```
TempTable:
LOAD RecNo() as InvID, * Inline [
```

```
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsName(2, InvDate, 4) AS MthsName
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve monthsname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	MthsName
28/03/2012	Nov-Dec 2012
10/12/2012	Jul-Aug 2013
5/2/2013	Sep-Oct 2013
31/3/2013	Nov-Dec2013
19/5/2013	Jan-Feb 2014
15/9/2013	May-Jun 2014
11/12/2013	Jul-Aug 2014
2/3/2014	Nov-Dec 2014
14/5/2014	Jan-Feb 2015
13/6/2014	Jan-Feb 2015
7/7/2014	Mar-Apr 2015
4/8/2014	Mar-Apr 2015

monthsstart

Bu fonksiyon bir taban tarih içeren ayın, iki aylık dönemin, çeyreğin, tersiyelin veya yarım yılın ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Önceki ve sonraki bir zaman dönemi için zaman damgasını bulmak da mümkündür.

Söz Dizimi:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
n_months	Dönemi tanımlayan ayların sayısı. Şunlardan biri olması gereken bir tamsayı veya bir tamsayıya çözümlenen bir ifade: 1 (inmonth() fonksiyonuna eşdeğer), 2 (iki aylık), 3 (inquarter() fonksiyonuna eşdeğer), 4 (dört aylık), or 6 (altı aylık).
date	Değerlendirilecek tarih.
period_no	Dönem **period_no** ile kaydırılabilir. Bu değer bir tamsayı ya da tamsayıya çözümlenen bir ifadedir ve burada 0 değeri **base_date** içeren dönemi belirtir. **period_no** içindeki negatif değerler önceki dönemleri; pozitif değerler ise sonraki dönemleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`monthsstart(4, '19/10/2013')`	1/09/2013 döndürür.
`monthsstart(4, '19/10/2013, -1)`	01/05/2013 döndürür.
`monthsstart(4, '19/10/2013', 0, 2 )`	01/10/2013 döndürür. Çünkü yılın başlangıcı 2. ay olur.

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, her bir fatura tarihi için iki aylık dönemin ilk gününü bulur (iki aylık bir dönem kadar ileri kaydırılmış olarak).

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
```

```
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsStart(2, InvDate, 1) AS BiMthsStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve MonthsStart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	BiMthsStart
28/03/2012	01/05/2012
10/12/2012	01/01/2013
5/2/2013	01/03/2013
31/3/2013	01/05/2013
19/5/2013	01/07/2013
15/9/2013	01/11/2013
11/12/2013	01/01/2014
2/3/2014	01/05/2014
14/5/2014	01/07/2014
13/6/2014	01/07/2014
7/7/2014	01/09/2014
4/8/2014	01/09/2014

monthstart

Bu fonksiyon, **date** içeren ayın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
MonthStart(date[, period_no])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, 0 olur ya da atlanırsa **date** içeren ayı belirtir. **period_no** içindeki negatif değerler önceki ayları; pozitif değerler ise sonraki ayları gösterir.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`monthstart('19/10/2001')`	01/10/2001 döndürür.
`monthstart('19/10/2001', -1)`	01/09/2001 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin ayı içinde ilk günü bulur. Burada base_date öğesi, period_no değerinin 4 olarak belirtilmesiyle dört hafta kaydırılmaktadır.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthStart(InvDate, 4) AS MthStart
```

```
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve monthstart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	MthStart
28/03/2012	01/07/2012
10/12/2012	01/04/2013
5/2/2013	01/06/2013
31/3/2013	01/07/2013
19/5/2013	01/09/2013
15/9/2013	01/01/2014
11/12/2013	01/04/2014
2/3/2014	01/07/2014
14/5/2014	01/09/2014
13/6/2014	01/10/2014
7/7/2014	01/11/2014
4/8/2014	01/12/2014

networkdays

networkdays fonksiyonu, isteğe bağlı olarak listelenen tüm **holiday** öğelerini dikkate alarak, **start_date** ve **end_date** arasındaki ve bu tarihleri de içeren iş günlerinin (Pazartesi - Cuma) sayısını döndürür.

Söz Dizimi:

```
networkdays (start_date, end_date [, holiday])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
start_date	Değerlendirilecek başlangıç tarihi.
end_date	Değerlendirilecek bitiş tarihi.

Bağımsız Değişken	Açıklama
holiday	İş günlerinden hariç tutulacak tatil dönemleri. Tatil dönemi, virgülle ayrılan bir başlangıç tarihi ve bir bitiş tarihi olarak belirtilir. **Örnek:** `'25/12/2013', '26/12/2013'` Virgüllerle ayrılmış olarak birden fazla tatil dönemi belirtebilirsiniz. **Örnek:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`networkdays ('19/12/2013', '07/01/2014')`	14 döndürür. Bu örnek tatil günlerini hesaba katmaz.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013')`	12 döndürür. Bu örnek 25/12/2013 ile 26/12/2013 arasındaki tatili hesaba katar.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014')`	10 döndürür. Bu örnek iki tatil dönemini hesaba katar.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
PayTable:
LOAD recno() as InvID, * INLINE [
InvRec|InvPaid
28/03/2012|28/04/2012
10/12/2012|01/01/2013
5/2/2013|5/3/2013
31/3/2013|01/5/2013
19/5/2013|12/6/2013
15/9/2013|6/10/2013
11/12/2013|12/01/2014
2/3/2014|2/4/2014
14/5/2014|14/6/2014
13/6/2014|14/7/2014
7/7/2014|14/8/2014
4/8/2014|4/9/2014
] (delimiter is '|');
NrDays:
Load *,
NetWorkDays(InvRec,InvPaid) As PaidDays
```

```
Resident PayTable;
Drop table PayTable;
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen NetworkDays değerlerini gösterir.

Sonuçlar tablosu

InvID	InvRec	InvPaid	PaidDays
1	28/03/2012	28/04/2012	23
2	10/12/2012	01/01/2013	17
3	5/2/2013	5/3/2013	21
4	31/3/2013	01/5/2013	23
5	19/5/2013	12/6/2013	18
6	15/9/2013	6/10/2013	15
7	11/12/2013	12/01/2014	23
8	2/3/2014	2/4/2014	23
9	14/5/2014	14/6/2014	23
10	13/6/2014	14/7/2014	22
11	7/7/2014	14/8/2014	29
12	4/8/2014	4/9/2014	24

now

Bu fonksiyon, sistem saatinden geçerli zamanın zaman damgasını döndürür. Varsayılan değer 1'dir.

Söz Dizimi:

```
now([ timer_mode])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timer_mode	Aşağıdaki değerleri alabilir: 0 (son tamamlanan veri yüklemesi zamanı) 1 (fonksiyon çağrısı zamanı) 2 (uygulamanın açıldığı zaman) *Fonksiyonu bir veri kod dosyasında kullanırsanız, **timer_mode=0** son bitirilen veri yüklemesinin zamanını sonuç olarak verirken **timer_mode=1** geçerli veri yüklemesinde fonksiyonun çağrılma zamanını verir.*

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`now( 0)`	Son veri yüklemesinin tamamlandığı zamanı döndürür.
`now( 1)`	Bir görselleştirme ifadesinde kullanıldığında, fonksiyon çağırma zamanını döndürür. Bir veri kod dosyasında kullanıldığında, geçerli veri yüklemesindeki fonksiyon çağırma zamanını döndürür.
`now( 2)`	Uygulamanın açıldığı zamanı döndürür.

quarterend

Bu fonksiyon, **date** içeren çeyreğin son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren çeyreği belirtir. **period_no** içindeki negatif değerler önceki çeyrekleri; pozitif değerler ise sonraki çeyrekleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`quarterend('29/10/2005')`	31/12/2005 23:59:59 döndürür.
`quarterend('29/10/2005', -1)`	30/09/2005 23:59:59 döndürür.
`quarterend('29/10/2005', 0, 3)`	30/11/2005 23:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin yer aldığı çeyrek içinde son günü bulur. Burada yılın ilk ayı 3. ay olarak belirtilmektedir.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
```

```
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
QuarterEnd(InvDate, 0, 3) AS QtrEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve quarterend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	QtrEnd
28/03/2012	31/05/2012
10/12/2012	28/02/2013
5/2/2013	28/02/2013
31/3/2013	31/05/2013
19/5/2013	31/05/2013
15/9/2013	30/11/2013
11/12/2013	28/02/2014
2/3/2014	31/05/2014
14/5/2014	31/05/2014
13/6/2014	31/08/2014
7/7/2014	31/08/2014
4/8/2014	31/08/2014

quartername

Bu fonksiyon, çeyreğin aylarını (**MonthNames** kod değişkenine göre biçimlendirilmiş) ve yılı, çeyreğin ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle gösteren bir görüntü değeri döndürür.

Söz Dizimi:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren çeyreği belirtir. **period_no** içindeki negatif değerler önceki çeyrekleri; pozitif değerler ise sonraki çeyrekleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`quartername('29/10/2013')`	Oct-Dec 2013 döndürür.
`quartername('29/10/2013', -1)`	Jul-Sep 2013 döndürür.
`quartername('29/10/2013', 0, 3)`	Sep-Nov 2013 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnekte, tablodaki her bir fatura tarihi için çeyrek adı *InvID* içeren çeyreğe dayalı olarak oluşturulur. Yılın ilk ayı, 4. ay olarak belirtilir.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

```
InvoiceData:
LOAD *,
QuarterName(InvDate, 0, 4) AS QtrName
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve quartername() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	QtrName
28/03/2012	Jan-Mar 2011
10/12/2012	Oct-Dec 2012
5/2/2013	Jan-Mar 2012
31/3/2013	Jan-Mar 2012
19/5/2013	Apr-Jun 2013
15/9/2013	Jul-Sep 2013
11/12/2013	Oct-Dec 2013
2/3/2014	Jan-Mar 2013
14/5/2014	Apr-Jun 2014
13/6/2014	Apr-Jun 2014
7/7/2014	Jul-Sep 2014
4/8/2014	Jul-Sep 2014

quarterstart

Bu fonksiyon, **date** içeren çeyreğin ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.

Bağımsız Değişken	Açıklama
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren çeyreği belirtir. **period_no** içindeki negatif değerler önceki çeyrekleri; pozitif değerler ise sonraki çeyrekleri gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`quarterstart('29/10/2005')`	01/10/2005 döndürür.
`quarterstart('29/10/2005', -1 )`	01/07/2005 döndürür.
`quarterstart('29/10/2005', 0, 3)`	01/09/2005 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin yer aldığı çeyrek içinde ilk günü bulur. Burada yılın ilk ayı 3. ay olarak belirtilmektedir.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
QuarterStart(InvDate, 0, 3) AS QtrStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve quarterstart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	QtrStart
28/03/2012	01/03/2012
10/12/2012	01/12/2012
5/2/2013	01/12/2012
31/3/2013	01/03/2013
19/5/2013	01/03/2013
15/9/2013	01/09/2013
11/12/2013	01/12/2013
2/3/2014	01/03/2014
14/5/2014	01/03/2014
13/6/2014	01/06/2014
7/7/2014	01/06/2014
4/8/2014	01/06/2014

second

Bu fonksiyon, **expression** öğesinin kesri standart sayı yorumlamasına göre saat olarak yorumlandığında, saniyeyi temsil eden bir tamsayı döndürür.

Söz Dizimi:

```
second (expression)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`second( '09:14:36' )`	36 döndürür
`second( '0.5555' )`	55 sonucunu döndürür (Çünkü 0,5555 = 13:19:55)

setdateyear

Bu fonksiyon, giriş olarak bir **timestamp** ve **year** alır ve **timestamp** öğesini girişte belirtilen **year** ile günceller.

Söz Dizimi:

```
setdateyear (timestamp, year)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Standart bir Qlik Sense zaman damgası (çoğu zaman sadece bir tarih).
year	Dört haneli yıl.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`setdateyear ('29/10/2005', 2013)`	'29/10/2013' döndürür
`setdateyear ('29/10/2005 04:26:14', 2013)`	'29/10/2013 04:26:14' döndürür Bir görselleştirmede zaman damgasının zaman bölümünü görmek için sayı biçimlendirmeyi Tarih olarak ayarlamanız ve Biçimlendirme için zaman değerlerini gösteren bir değer seçmeniz gerekir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
```

```
];
```

Sonuçta ortaya çıkan tablo orijinal tarihleri ve yılın 2013 olarak ayarlandığı bir sütunu içerir.

Sonuçlar tablosu

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

Bu fonksiyon, giriş olarak bir **timestamp**, **month** ve **year** alır ve **timestamp** öğesini girişte belirtilen **year** ve **month** ile günceller. .

Söz Dizimi:

```
SetDateYearMonth (timestamp, year, month)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Standart bir Qlik Sense zaman damgası (çoğu zaman sadece bir tarih).
year	Dört haneli yıl.
month	Bir veya iki haneli ay.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`setdateyearmonth ('29/10/2005', 2013, 3)`	'29/03/2013' döndürür
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	'29/03/2013 04:26:14' döndürür Bir görselleştirmede zaman damgasının zaman bölümünü görmek için sayı biçimlendirmeyi Tarih olarak ayarlamanız ve Biçimlendirme için zaman değerlerini gösteren bir değer seçmeniz gerekir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta ortaya çıkan tablo orijinal tarihleri ve yılın 2013 olarak ayarlandığı bir sütunu içerir.

Sonuçlar tablosu

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

Bu fonksiyon, Windows'da tanımlandığı şekilde, geçerli saat diliminin adını döndürür.

Söz Dizimi:

```
TimeZone( )
```

Dönüş verileri türü: dize

Örnek:

```
timezone( )
```

today

Bu fonksiyon, sistem saatinden geçerli tarihi döndürür.

Söz Dizimi:

```
today([ timer_mode])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timer_mode	Aşağıdaki değerleri alabilir: 0 (son tamamlanan veri yüklemesi günü) 1 (fonksiyon çağrısı günü) 2 (uygulamanın açıldığı gün) *Fonksiyonu bir veri kod dosyasında kullanırsanız, **timer_mode=0** son bitirilen veri yüklemesinin gününü sonuç olarak verirken **timer_mode=1** geçerli veri yüklemesinin gününü verir.*

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`Today( 0)`	Son bitirilen veri yüklemesinin gününü döndürür.
`Today( 1)`	Bir görselleştirme ifadesinde kullanıldığında, fonksiyon çağırma gününü döndürür. Veri kod dosyasında kullanıldığında, geçerli veri yüklemesinin başladığı günü döndürür.
`Today( 2)`	Uygulamanın açıldığı günü döndürür.

UTC

Geçerli Coordinated Universal Time değerini döndürür.

Söz Dizimi:

```
UTC( )
```

Dönüş verileri türü: dual

Örnek:

```
utc( )
```

week

Bu fonksiyon, ISO 8601 uyarınca hafta numarasını temsil eden bir tamsayı döndürür. Hafta numarası, standart sayı yorumlamasına göre ifadenin tarih yorumlamasından hesaplanır.

Söz Dizimi:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Bir zaman damgası olarak değerlendirilecek veya bir zaman damgasına çözümlenen bir ifade olarak dönüştürülecek tarih; örneğin: '2012-10-12'.

Bağımsız Değişken	Açıklama
first_week_day	**first_week_day** öğesini belirtmezseniz **FirstWeekDay** değişkeninin değeri haftanın ilk günü olarak kullanılır. Haftanın ilk günü olarak başka bir günü kullanmak istiyorsanız **first_week_day** ayarını şöyle yapın: • Pazartesi için 0 • Salı için 1 • Çarşamba için 2 • Perşembe için 3 • Cuma için 4 • Cumartesi için 5 • Pazar için 6 Fonksiyonun döndürdüğü tamsayı artık, **first_week_day** ile ayarladığınız haftanın ilk gününü kullanır.
broken_weeks	**broken_weeks** öğesini belirtmezseniz **BrokenWeeks** değişkeninin değeri, haftaların bölünmüş olup olmadığını tanımlamak için kullanılır. Varsayılan olarak, Qlik Sense fonksiyonları bölünmemiş haftaları kullanır. Bunun anlamı şudur: • Bazı yıllarda 1. hafta Aralık ayı içinde başlar ve bazı yıllarda 52. veya 53. hafta Ocak ayına devam eder. • 1. haftanın Ocak ayı içinde her zaman en az 4 günü vardır. Bunun alternatifi bölünmüş haftaları kullanmaktır. • 52. veya 53. hafta Ocak ayına devam etmez. • 1. hafta 1 Ocak'tan itibaren başlar ve çoğu durumda tam bir hafta değildir. Aşağıdaki değerler kullanılabilir: • 0 (=bölünmemiş haftaları kullan) • 1 (= bölünmüş haftaları kullan)

Bağımsız Değişken	Açıklama
reference_day	**reference_day** öğesini belirtmezseniz **ReferenceDay** değişkeninin değeri, 1. haftayı tanımlamak için Ocak ayındaki hangi günün referans gün olarak ayarlanacağını tanımlamak için kullanılır. Varsayılan olarak, Qlik Sense fonksiyonları referans gün olarak 4 kullanır. Bu da 1. haftanın 4 Ocak gününü içermesi gerektiği veya başka bir deyişle 1. haftanın Ocak ayında her zaman en az 4 günü olması gerektiği anlamına gelir. Farklı bir referans gün ayarlamak için aşağıdaki değerler kullanılabilir: • 1 (= 1 Ocak) • 2 (= 2 Ocak) • 3 (= 3 Ocak) • 4 (= 4 Ocak) • 5 (= 5 Ocak) • 6 (= 6 Ocak) • 7 (= 7 Ocak)

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`week( '2012-10-12' )`	41 döndürür.
`week( '35648' )`	35648 = 1997-08-06 olduğundan 32 döndürür
`week('2012-10-12', 0, 1)`	42 döndürür

weekday

Bu fonksiyon şunları içeren bir ikili değer döndürür:

- **DayNames** ortam değişkeninde tanımlanan bir gün adı.
- Haftanın nominal gününe karşılık gelen 0-6 arasında bir tamsayı (0-6).

Söz Dizimi:

```
weekday(date [,first_week_day=0])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
first_week_day	**first_week_day** öğesini belirtmezseniz **FirstWeekDay** değişkeninin değeri haftanın ilk günü olarak kullanılır. Haftanın ilk günü olarak başka bir günü kullanmak istiyorsanız **first_week_day** ayarını şöyle yapın: • Pazartesi için 0 • Salı için 1 • Çarşamba için 2 • Perşembe için 3 • Cuma için 4 • Cumartesi için 5 • Pazar için 6 Fonksiyonun döndürdüğü tamsayı artık, **first_week_day** ile ayarladığınız haftanın ilk gününü taban (0) olarak kullanır. *FirstWeekDay (page 153)*

Örnek: Grafik ifadesi
Aksi belirtilmedikçe **FirstWeekDay** bu örneklerde 0 olarak ayarlanır.

Kod örnekleri

Örnek	Sonuç
`weekday( '1971-10-12' )`	'Sal' ve 1 döndürür
`weekday( '1971-10-12' , 6)`	'Sal' ve 2 döndürür. Bu örnekte Pazar (6) gününü haftanın ilk günü olarak kullanıyoruz.

`SET FirstWeekDay = 6;`
 	
`...`
`weekday( '1971-10-12')`
	'Sal' ve 2 döndürür.

Örnek: Yükleme kodu

Yükleme kodu

FirstWeekDay ve *ReferenceDay* önceden kodda ayarlanmış olsa da haftanın gününü temsil eden bir sayı ve dize döndürmek için yükleme kodunda *weekday* kullanılabilir. Aşağıdaki yükleme kodu, *FirstWeekDay* ve *ReferenceDay* değerlerini içerir ve sonra *transaction_date* sütunundaki verilerden haftanın günlerini temsil eden dizeleri ve sayıları döndürmek için *weekday* öğesini kullanır.

Gösterilen sonuçlarda *Day* sütunu, döndürülen dizeleri içerirken *Numeric value of Day* ve *Numeric value of week starting from Sunday* ise döndürülen sayısal değerleri içerir. Döndürülen veri türünün sayısal olduğundan emin olmanın kolay bir yolu olarak yükleme kodunda *weekday*, 1 ile çarpılır.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
Load
*,
WeekDay(transaction_date) as [Day],
1*WeekDay(transaction_date) as [Numeric value of Day]
1*WeekDay(transaction_date, 6) as [Numeric value of a week starting from Sunday],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Sonuçlar

Yükleme kodunda kullanılmakta olan weekday fonksiyonunun sonuçlarını gösteren Qlik Sense tablosu.

transaction_id	transaction_ date	Gün	Gün sayısal değeri	Pazar ile başlayan bir hafta için sayısal değer
3750	20180830	Thu	3	4
3751	20180907	Thu	3	4
3752	20180916	Sat	5	6
3753	20180922	Fri	4	5
3754	20180922	Fri	4	5
3756	20180922	Fri	4	5
3757	20180923	Sat	5	6

weekend

Bu fonksiyon, **date** öğesini içeren takvim haftasının son günün (Pazar) son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi, kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
WeekEnd(date [, period_no[, first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**shift** bir tamsayı olup, burada 0 değeri **date** içeren haftayı belirtir. Shift içindeki negatif değerler önceki haftaları; pozitif değerler ise sonraki haftaları gösterir.

Bağımsız Değişken	Açıklama
first_week_day	Haftanın başladığı günü belirtir. Atlandığı takdirde, **FirstWeekDay** değişkeninin değeri kullanılır. Olası **first_week_day** değerleri şunlardır: • Pazartesi için 0 • Salı için 1 • Çarşamba için 2 • Perşembe için 3 • Cuma için 4 • Cumartesi için 5 • Pazar için 6 *FirstWeekDay (page 153)*

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Örnek	Sonuç
`weekend('10/01/2013')`	12/01/2013 23:59:59 döndürür.
`weekend('10/01/2013', -1)`	06/01/2013 23:59:59. döndürür
`weekend('10/01/2013', 0, 1)`	14/01/2013 23:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin haftasından sonraki haftanın son gününü bulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

```
InvoiceData:
LOAD *,
WeekEnd(InvDate, 1) AS WkEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve weekend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	WkEnd
28/03/2012	08/04/2012
10/12/2012	23/12/2012
5/2/2013	17/02/2013
31/3/2013	07/04/2013
19/5/2013	26/05/2013
15/9/2013	22/09/2013
11/12/2013	22/12/2013
2/3/2014	09/03/2014
14/5/2014	25/05/2014
13/6/2014	22/06/2014
7/7/2014	20/07/2014
4/8/2014	17/08/2014

weekname

Bu fonksiyon, **date** öğesini içeren haftanın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle yıl ve hafta sayısını gösteren bir değer döndürür.

Söz Dizimi:

```
WeekName(date[, period_no[,first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**shift** bir tamsayı olup, burada 0 değeri **date** içeren haftayı belirtir. Shift içindeki negatif değerler önceki haftaları; pozitif değerler ise sonraki haftaları gösterir.
first_week_day	Haftanın başladığı günü belirtir. Atlandığı takdirde, **FirstWeekDay** değişkeninin değeri kullanılır. Olası **first_week_day** değerleri şunlardır: • Pazartesi için 0 • Salı için 1 • Çarşamba için 2 • Perşembe için 3 • Cuma için 4 • Cumartesi için 5 • Pazar için 6 *FirstWeekDay (page 153)*

Örnekler ve sonuçlar:

Örnek	Sonuç
`weekname('12/01/2013')`	2013/02 döndürür.
`weekname('12/01/2013', -1)`	2013/01 döndürür.
`weekname('12/01/2013', 0, 1)`	'2013/02 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnekte, tablodaki her bir fatura tarihi için, haftanın içinde yer aldığı yıldan ve period_no değerinin 1 olarak belirtilmesiyle bir hafta kaydırılmış ilişkili hafta sayısından haftanın adı oluşturulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
```

```
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekName(InvDate, 1) AS WkName
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve weekname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	WkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/29
4/8/2014	2014/33

weekstart

Bu fonksiyon, **date** içeren takvim haftasının ilk gününün (Pazartesi) ilk milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
WeekStart(date [, period_no[, first_week_day]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**shift** bir tamsayı olup, burada 0 değeri **date** içeren haftayı belirtir. Shift içindeki negatif değerler önceki haftaları; pozitif değerler ise sonraki haftaları gösterir.
first_week_day	Haftanın başladığı günü belirtir. Atlandığı takdirde, **FirstWeekDay** değişkeninin değeri kullanılır. Olası **first_week_day** değerleri şunlardır: • Pazartesi için 0 • Salı için 1 • Çarşamba için 2 • Perşembe için 3 • Cuma için 4 • Cumartesi için 5 • Pazar için 6 *FirstWeekDay (page 153)*

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`weekstart('12/01/2013')`	07/01/2013 döndürür.
`weekstart('12/01/2013', -1 )`	31/11/2012 döndürür.
`weekstart('12/01/2013', 0, 1)`	08/01/2013 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin haftasından sonraki haftanın ilk gününü bulur.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
```

```
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekStart(InvDate, 1) AS WkStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve weekstart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	WkStart
28/03/2012	02/04/2012
10/12/2012	17/12/2012
5/2/2013	11/02/2013
31/3/2013	01/04/2013
19/5/2013	20/05/2013
15/9/2013	16/09/2013
11/12/2013	16/12/2013
2/3/2014	03/03/2014
14/5/2014	19/05/2014
13/6/2014	16/06/2014
7/7/2014	14/07/2014
4/8/2014	11/08/2014

weekyear

Bu fonksiyon, ISO 8601 uyarınca hafta numarasının ait olduğu yılı döndürür. Hafta sayısı, 1 ve yaklaşık 52 arasında değişir.

Söz Dizimi:

```
weekyear(expression)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`weekyear( '1996-12-30' )`	1997 yılının 1. haftası 30.12.1996 tarihinde başladığından 1997 sonucunu döndürür
`weekyear( '1997-01-02' )`	1997 döndürür
`weekyear( '1997-12-28' )`	1997 döndürür
`weekyear( '1997-12-30' )`	1998 yılının 1. haftası 29.12.1997 tarihinde başladığından 1998 sonucunu döndürür
`weekyear( '1999-01-02' )`	1998 yılının 53. haftası 03.01.1999 tarihinde sona erdiğinden 1998 sonucunu döndürür

Sınırlamalar:

Bazı yıllarda 1. hafta Aralık ayında başlar (örn. Aralık 1997). Bazı yıllar ise önceki yılın 53. haftasıyla başlar (örn. Ocak 1999). Hafta sayısının başka bir yıla ait olduğu bu birkaç gün için, **year** ve **weekyear** fonksiyonları farklı değerler döndürür.

year

Bu fonksiyon, **expression** öğesi standart sayı yorumlamasına göre tarih olarak yorumlandığında, yılı temsil eden bir tamsayı döndürür.

Söz Dizimi:

```
year(expression)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`year( '2012-10-12' )`	2012 döndürür
`year( '35648' )`	35648 = 1997-08-06 olduğundan 1997 döndürür

yearend

Bu fonksiyon, **date** içeren yılın son gününün son milisaniyesinin zaman damgasına karşılık gelen bir değer döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren yılı belirtir. **period_no** içindeki negatif değerler önceki yılları; pozitif değerler ise sonraki yılları gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`yearend ( '19/10/2001' )`	31/12/2001 23:59:59 döndürür.
`yearend ( '19/10/2001', -1 )`	31/12/2000 23:59:59 döndürür.
`yearend ( '19/10/2001', 0, 4)`	31/03/2002 23:59:59 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin yer aldığı yıl içinde son günü bulur. Burada yılın ilk ayı 4. ay olarak belirtilmektedir.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
```

```
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
YearEnd(InvDate, 0, 4) AS YrEnd
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve yearend() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	YrEnd
28/03/2012	31/03/2011
10/12/2012	31/03/2012
5/2/2013	31/03/2013
31/3/2013	31/03/2013
19/5/2013	31/03/2014
15/9/2013	31/03/2014
11/12/2013	31/03/2014
2/3/2014	31/03/2014
14/5/2014	31/03/2015
13/6/2014	31/03/2015
7/7/2014	31/03/2015
4/8/2014	31/03/2015

yearname

Bu fonksiyon, **date** öğesini içeren yılın ilk gününün ilk milisaniyesinin zaman damgasına karşılık gelen bir temel sayısal değerle görüntü değeri olarak dört basamaklı bir yıl döndürür.

Söz Dizimi:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren yılı belirtir. **period_no** içindeki negatif değerler önceki yılları; pozitif değerler ise sonraki yılları gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin. Bu durumda, görüntü değeri iki yılı gösteren bir dize olur.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`yearname ( '19/10/2001')`	2001 döndürür.
`yearname ( '19/10/2001', -1 )`	'2000 döndürür.
`yearname ( '19/10/2001', 0, 4)`	'2001-2002 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin yer aldığı yıl içinde ilk günü bulur. Burada yılın ilk ayı 4. ay olarak belirtilmektedir.

Bu örnek, tablodaki her bir fatura tarihinin içinde bulunduğu yıllar için dört+dört haneli bir ad oluşturur. Bunun nedeni yılın ilk ayının 4. ay olarak belirtilmiş olmasıdır.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
```

```
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
YearName(InvDate, 0, 4) AS YrName
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve yearname() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır.

Sonuçlar tablosu

InvDate	YrName
28/03/2012	2011-2012
10/12/2012	2012-2013
5/2/2013	2012-2013
31/3/2013	2012-2013
19/5/2013	2013-2014
15/9/2013	2013-2014
11/12/2013	2013-2014
2/3/2014	2013-2014
14/5/2014	2014-2015
13/6/2014	2014-2015
7/7/2014	2014-2015
4/8/2014	2014-2015

yearstart

Bu fonksiyon, **date** içeren yılın ilk gününün başlangıcına karşılık gelen bir zaman damgası döndürür. Varsayılan çıktı biçimi kodda ayarlanan **DateFormat** olur.

Söz Dizimi:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
date	Değerlendirilecek tarih.
period_no	**period_no** bir tamsayı olup, burada 0 değeri **date** içeren yılı belirtir. **period_no** içindeki negatif değerler önceki yılları; pozitif değerler ise sonraki yılları gösterir.
first_month_of_year	Ocak'ta başlamayan (mali) yıllarla çalışmak istiyorsanız, **first_month_of_year** içinde 2 ile 12 arasında bir değer belirtin.

Örnekler ve sonuçlar:
Bu örnekler **DD/MM/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyanızın en üstündeki **SET DateFormat** deyiminde belirtilir. Örneklerdeki biçimi gereksinimlerinize uyacak şekilde değiştirin.

Kod örnekleri

Örnek	Sonuç
`yearstart ('19/10/2001')`	01/01/2001 döndürür.
`yearstart ('19/10/2001', -1)`	01/01/2000 döndürür.
`yearstart ('19/10/2001', 0, 4)`	01/04/2001 döndürür.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

Bu örnek, tablodaki her bir fatura tarihinin yer aldığı yıl içinde ilk günü bulur. Burada yılın ilk ayı 4. ay olarak belirtilmektedir.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
```

```
4/8/2014
];

InvoiceData:
LOAD *,
YearStart(InvDate, 0, 4) AS YrStart
Resident TempTable;
Drop table TempTable;
```

Sonuçta ortaya çıkan tabloda orijinal tarihler ve yearstart() fonksiyonunun döndürdüğü değeri içeren bir sütun yer alır. Özellikler panelinde biçimlendirmeyi belirterek tam zaman damgasını görüntüleyebilirsiniz.

Sonuçlar tablosu

InvDate	YrStart
28/03/2012	01/04/2011
10/12/2012	01/04/2012
5/2/2013	01/04/2012
31/3/2013	01/04/2012
19/5/2013	01/04/2013
15/9/2013	01/04/2013
11/12/2013	01/04/2013
2/3/2014	01/04/2013
14/5/2014	01/04/2014
13/6/2014	01/04/2014
7/7/2014	01/04/2014
4/8/2014	01/04/2014

yeartodate

Bu fonksiyon giriş zaman damgasının kodun yüklendiği yılda olup olmadığını bulur ve bu yıldaysa True, değilse False değerini döndürür.

Söz Dizimi:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Dönüş verileri türü: Boole

İsteğe bağlı parametrelerden hiçbiri kullanılmazsa, yıl içinde belirli bir tarihe kadar, 1 Ocak'tan son kod yürütme tarihine kadar ve bu tarihi de içerecek şekilde, bir takvim yılı dahilindeki herhangi bir tarih anlamına gelir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
timestamp	Değerlendirilecek zaman damgası (örneğin, '2012-10-12').
yearoffset	Bir **yearoffset** belirtilmesiyle, **yeartodate** başka bir yıldaki aynı dönem için True değerini döndürür. Negatif **yearoffset** önceki bir yılı belirtirken, pozitif kayma gelecekteki bir yılı belirtir. En yeni year-to-date yearoffset = -1 olarak belirtilmesiyle elde edilir. Atlandığı takdirde 0 olduğu varsayılır.
firstmonth	1 ile 12 arasında bir **firstmonth** belirtildiğinde (atlandığı takdirde 1) yılın başlangıcı herhangi bir ayın ilk gününe ileri taşınabilir. Örneğin, 1 Mayıs'ta başlayan bir mali yıl ile çalışmak istiyorsanız **firstmonth** = 5 olarak belirtin.
todaydate	Bir **todaydate** belirtildiğinde (atlandığı takdirde son kod yürütme işleminin zaman damgası), dönemin üst sınırı olarak kullanılan günü taşımak mümkündür.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde son yeniden yükleme zamanının = 2011-11-18 olduğu varsayılır

Kod örnekleri

Örnek	Sonuç
`yeartodate( '2010-11-18')`	şunu döndürür: False
`yeartodate( '2011-02-01')`	şunu döndürür: True
`yeartodate( '2011-11-18')`	şunu döndürür: True
`yeartodate( '2011-11-19')`	şunu döndürür: False
`yeartodate( '2011-11-19', 0, 1, '2011-12-31')`	şunu döndürür: True
`yeartodate( '2010-11-18', -1)`	şunu döndürür: True
`yeartodate( '2011-11-18', -1)`	şunu döndürür: False
`yeartodate( '2011-04-30', 0, 5)`	şunu döndürür: False
`yeartodate( '2011-05-01', 0, 5)`	şunu döndürür: True

5.8 Üstel ve logaritmik fonksiyonlar

Bu bölümde, üstel ve logaritmik hesaplamalarla ilgili fonksiyonlar açıklanmaktadır. Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

Aşağıdaki fonksiyonlarda parametreler, **x** ve **y** değerlerinin gerçek değerli sayılar olarak yorumlanması gereken ifadelerdir.

exp

e doğal logaritmasının taban olarak kullanıldığı doğal üstel fonksiyon e^x. Sonuç pozitif bir sayıdır.

```
exp(x )
```

Örnekler ve sonuçlar:

`exp(3)`, 20,085 değerini döndürür.

log

x değerinin doğal logaritması. Fonksiyon yalnızca **x**> 0 olması durumunda tanımlanır. Sonuç bir sayıdır.

```
log(x )
```

Örnekler ve sonuçlar:

`log(3)`, 1,0986 değerini döndürür

log10

x değerinin bayağı logaritması (10 tabanlı). Fonksiyon yalnızca **x**> 0 olması durumunda tanımlanır. Sonuç bir sayıdır.

```
log10(x )
```

Örnekler ve sonuçlar:

`log10(3)`, 0,4771 değerini döndürür

pow

x değerinin **y**. kuvvetini döndürür. Sonuç bir sayıdır.

```
pow(x,y )
```

Örnekler ve sonuçlar:

`pow(3, 3)`, 27 değerini döndürür

sqr

x kare (**x** değerinin 2. kuvveti). Sonuç bir sayıdır.

```
sqr (x )
```

Örnekler ve sonuçlar:

`sqr(3)`, 9 değerini döndürür

sqrt

x değerinin kare kökü. Fonksiyon yalnızca **x** >= 0 olması durumunda tanımlanır. Sonuç pozitif bir sayıdır.

```
sqrt(x )
```

Örnekler ve sonuçlar:

`sqrt(3)`, 1,732 değerini döndürür

5.9 Alan fonksiyonları

Bu fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir.

Alan fonksiyonları, farklı yönleriyle alan seçimlerini tanımlayan tamsayılar ya da dizeler döndürür.

Sayım fonksiyonları

GetAlternativeCount

GetAlternativeCount(), tanımlanan alanda alternatif (açık gri) değerlerin sayısını bulmak için kullanılır.

```
GetAlternativeCount - grafik fonksiyonu (field_name)
```

GetExcludedCount

GetExcludedCount() tanımlanan alandaki hariç tutulan benzersiz değerlerin sayısını bulur. Hariç tutulan değerler; alternatif (açık gri), hariç tutulan (koyu gri) ve seçili hariç tutulan (onay işaretiyle birlikte koyu gri) alanları içerir.

```
GetExcludedCount - grafik fonksiyonu (page 571)(field_name)
```

GetNotSelectedCount

Bu grafik fonksiyonu **fieldname** adlı alandaki seçili olmayan değerlerin sayısını döndürür. Bu fonksiyonun ilgili olabilmesi için alan and-modunda olmalıdır.

```
GetNotSelectedCount - grafik fonksiyonu(fieldname [, includeexcluded=false])
```

GetPossibleCount

GetPossibleCount(), tanımlanan alanda olası değerlerin sayısını bulmak için kullanılır. Tanımlanan alan seçimler içeriyorsa, seçili (yeşil) alanlar sayılır. Aksi takdirde ilişkili (beyaz) değerler sayılır.

```
GetPossibleCount - grafik fonksiyonu(field_name)
```

GetSelectedCount

GetSelectedCount(), bir alandaki seçili (yeşil) değerlerin sayısını bulur.

```
GetSelectedCount - grafik fonksiyonu (field_name [, include_excluded])
```

Alan ve seçim fonksiyonları

GetCurrentSelections

GetCurrentSelections(), uygulamadaki geçerli seçimlerin listesini döndürür. Seçimler bunun yerine arama kutusunda bir arama dizesi kullanılarak yapıldıysa **GetCurrentSelections()**, arama dizesini döndürür.

```
GetCurrentSelections - grafik fonksiyonu([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

GetFieldSelections(), bir alandaki geçerli seçimler ile bir **dize** döndürür.

```
GetFieldSelections - grafik fonksiyonu ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() boyutun adını döndürür. **Index**, döndürülmesi gereken boyutu belirten isteğe bağlı tamsayıdır.

```
GetObjectDimension - grafik fonksiyonu ([index])
```

GetObjectField

GetObjectField(), boyutun adını döndürür. **Index**, döndürülmesi gereken boyutu belirten isteğe bağlı bir tamsayıdır.

```
GetObjectField - grafik fonksiyonu ([index])
```

GetObjectMeasure

GetObjectMeasure(), hesaplamanın adını döndürür. **Index**, döndürülmesi gereken hesaplamayı belirten isteğe bağlı bir tamsayıdır.

```
GetObjectMeasure - grafik fonksiyonu ([index])
```

GetAlternativeCount - grafik fonksiyonu

GetAlternativeCount(), tanımlanan alanda alternatif (açık gri) değerlerin sayısını bulmak için kullanılır.

Söz Dizimi:

```
GetAlternativeCount (field_name)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Ölçülecek veri aralığını içeren alan.

Örnekler ve sonuçlar:

Aşağıdaki örnekte, bir filtre bölmesine yüklenen **First name** alanı kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde **John** seçildiği varsayılırsa. `GetAlternativeCount ([First name])`	4; çünkü **First name** içinde 4 benzersiz ve hariç tutulan (gri) değer vardır.
John ve **Peter** seçildiği varsayılırsa. `GetAlternativeCount ([First name])`	3; çünkü **First name** içinde 3 benzersiz ve hariç tutulan (gri) değer vardır.
First name içinde hiçbir değer seçilmediği varsayılırsa. `GetAlternativeCount ([First name])`	0; çünkü hiçbir seçim yoktur.

Örnekte kullanılan veriler:

```
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 - grafik fonksiyonu

GetCurrentSelections(), uygulamadaki geçerli seçimlerin listesini döndürür. Seçimler bunun yerine arama kutusunda bir arama dizesi kullanılarak yapıldıysa **GetCurrentSelections()**, arama dizesini döndürür.

Seçenekler kullanılırsa record_sep öğesini belirtmeniz gerekir. Yeni bir satır belirtmek için **record_sep** öğesini **chr(13)&chr(10)** olarak ayarlayın.

İkisi dışında tüm değerler ya da biri dışında tüm değerler seçilirse, sırasıyla 'NOT x,y' veya 'NOT y' biçimi kullanılır. Tüm değerleri seçerseniz ve tüm değerlerin sayımı max_values değerinden büyükse, ALL metni döndürülür.

Söz Dizimi:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
record_sep	Alan kayıtları arasına koyulması gereken ayırıcı. Varsayılan <CR><LF> değeri yeni bir satır anlamına gelir.
tag_sep	Alan adı etiketi ile alan değerleri arasına koyulması gereken ayırıcı. Varsayılan ': ' işaretidir.
value_sep	Alan değerleri arasına koyulacak ayırıcı. Varsayılan, ', ' işaretidir.
max_values	Ayrı ayrı listelenecek olan alan değerlerinin maksimum sayısıdır. Çok sayıda değer seçildiğinde, bunun yerine 'x/y değer' biçimi kullanılır. Varsayılan 6'dır.
state_name	Belirli bir görselleştirme için seçilen alternatif durumun adı. **state_name** bağımsız değişkeni kullanılırsa yalnızca belirtilen durum adıyla ilişkili seçimler hesaba katılır.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde, biri **First name** adı için ve biri de **Initials** için olmak üzere, farklı filtre bölmelerine yüklenen iki alan kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde **John** seçildiği varsayılırsa. `GetCurrentSelections ()`	'First name: John'
First name içinde **John** ve **Peter** seçildiği varsayılırsa. `GetCurrentSelections ()`	'First name: John, Peter'
First name içinde **John** ve **Peter** ve **Initials** içinde **JA** seçildiği varsayılırsa. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
First name içinde **John** ve **Initials** içinde **JA** seçildiği varsayılırsa. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
First name içinde Sue hariç tüm adları seçtiğiniz ve **Initials** içinde hiçbir seçim yapılmadığı varsayılırsa. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Örnekte kullanılan veriler:

```
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 - grafik fonksiyonu

GetExcludedCount() tanımlanan alandaki hariç tutulan benzersiz değerlerin sayısını bulur. Hariç tutulan değerler; alternatif (açık gri), hariç tutulan (koyu gri) ve seçili hariç tutulan (onay işaretiyle birlikte koyu gri) alanları içerir.

Söz Dizimi:

```
GetExcludedCount (field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
field_name	Ölçülecek veri aralığını içeren alan.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde biri **First name** için, biri **Last name** için ve biri de **Initials** için olmak üzere, farklı filtre bölmelerine yüklenen üç alan kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde bir değer seçilmediyse.	`GetExcludedCount (Initials) = 0` Bir seçim yoktur.
First name içinde **John** seçilirse.	`GetExcludedCount (Initials) = 5` Koyu gri renkli **Initials** bölümünde hariç tutulan 5 değer vardır. **First name** içinde John seçimi ile ilişkili olması nedeniyle altıncı hücre (JA) beyaz olacaktır.
John ve **Peter** seçilirse.	`GetExcludedCount (Initials) = 3` **Initials** içinde John, 1 değerle ilişkilidir ve Peter, 2 değerle ilişkilidir.
First name içinde **John** ve **Peter** seçilirse, **Last name** içinde **Franc** seçilir.	`GetExcludedCount ([First name]) = 4` Koyu gri renkli **First name** bölümünde hariç tutulan 4 değer vardır. Alternatif ve seçili hariç tutulan alanlar dahil olmak üzere, hariç tutulan değerler içeren alanlar için **GetExcludedCount()** değerlendirilir.

Örnekler	Sonuçlar
First name içinde **John** ve **Peter** seçilirse, **Last name** içinde **Franc** ve **Anderson** seçilir.	`GetExcludedCount (Initials) = 4` Koyu gri renkli **Initials** bölümünde hariç tutulan 4 değer vardır. Diğer iki hücre (JA ve PF), **First name** içinde John ve Peter seçimleriyle ilişkili olduğundan beyaz olacaktır.
First name içinde **John** ve **Peter** seçilirse, **Last name** içinde **Franc** ve **Anderson** seçilir.	`GetExcludedCount ([Last name]) = 4` **Initials** bölümünde hariç tutulan 4 değer vardır. Devonshire, açık gri renkteyken Brown, Carr ve Elliot ise koyu gri renktedir.

Örnekte kullanılan veriler:

```
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 - grafik fonksiyonu

GetFieldSelections(), bir alandaki geçerli seçimler ile bir **dize** döndürür.

Değerlerin ikisi dışında tümü ya da biri dışında tümü seçilirse, sırasıyla 'NOT x,y' veya 'NOT y' biçimi kullanılır. Tüm değerleri seçerseniz ve tüm değerlerin sayımı max_values değerinden büyükse, ALL metni döndürülür.

Söz Dizimi:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Dönüş verileri türü: dize

Döndürülen dize biçimleri

Biçim	Açıklama
'a, b, c'	Seçilen değerlerin sayısı max_values veya daha azsa, döndürülen dize seçilen değerlerin bir listesidir. Değerler sınırlayıcı olarak value_sep ile ayrılır.
'NOT a, b, c'	Seçilmeyen değerlerin sayısı max_values veya daha azsa, döndürülen dize seçilmeyen değerlerin öneki NOT olan bir listesidir. Değerler sınırlayıcı olarak value_sep ile ayrılır.

Biçim	Açıklama
'x of y'	x = seçilen değerlerin sayısı y = toplam değer sayısı Bu, max_values < x < (y - max_values) olduğunda döndürülür.
'ALL'	Tüm değerler seçildiyse döndürülür.
'-'	Hiçbir değer seçilmediyse döndürülür.
<search string>	Arama kullanarak seçim yaptıysanız, arama dizesi döndürülür.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
field_name	Ölçülecek veri aralığını içeren alan.
value_sep	Alan değerleri arasına koyulacak ayırıcı. Varsayılan, ', ' işaretidir.
max_values	Ayrı ayrı listelenecek olan alan değerlerinin maksimum sayısıdır. Çok sayıda değer seçildiğinde, bunun yerine 'x/y değer' biçimi kullanılır. Varsayılan 6'dır.
state_name	Belirli bir görselleştirme için seçilen alternatif durumun adı. **state_name** bağımsız değişkeni kullanılırsa yalnızca belirtilen durum adıyla ilişkili seçimler hesaba katılır.

Örnekler ve sonuçlar:

Aşağıdaki örnekte, bir filtre bölmesine yüklenen **First name** alanı kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde **John** seçildiği varsayılırsa. `GetFieldSelections ([First name])`	'John'
John ve **Peter** seçildiği varsayılırsa. `GetFieldSelections ([First name])`	'John,Peter'

Örnekler	Sonuçlar
John ve **Peter** seçildiği varsayılırsa. `GetFieldSelections ([First name],'; ')`	'John; Peter'
First name içinde **John**, **Sue**, **Mark** seçildiği varsayılırsa. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter'; çünkü max_values bağımsız değişkeninin değeri olarak 2 değeri belirtilmektedir. Aksi takdirde, sonuç John; Sue; Mark. olurdu.

Örnekte kullanılan veriler:

```
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 - grafik fonksiyonu

Bu grafik fonksiyonu **fieldname** adlı alandaki seçili olmayan değerlerin sayısını döndürür. Bu fonksiyonun ilgili olabilmesi için alan and-modunda olmalıdır.

Söz Dizimi:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fieldname	Değerlendirilecek alanın adı.
includeexcluded	**includeexcluded** True olarak belirtilirse, sayım başka bir alandaki seçimler tarafından hariç tutulan seçilen değerleri içerir.

Örnekler:

```
GetNotSelectedCount( Country )
GetNotSelectedCount( Country, true )
```

GetObjectDimension - grafik fonksiyonu

GetObjectDimension() boyutun adını döndürür. **Index**, döndürülmesi gereken boyutu belirten isteğe bağlı tamsayıdır.

Şu konumlarda bir grafikte bu işlevi kullanamazsınız: başlık, alt başlık, alt bilgi, referans çizgisi ifadesi.

Object ID kullanarak bir boyutun veya hesaplamanın adını başka bir nesnede referans veremezsiniz.

Söz Dizimi:

```
GetObjectDimension ([index])
```

Örnek:

```
GetObjectDimension(1)
```

Örnek: Grafik ifadesi

Bir grafik ifadesinde GetObjectDimension fonksiyonunun örneklerini gösteren Qlik Sense tablosu

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
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Hesaplama adının döndürülmesini istiyorsanız **GetObjectMeasure** fonksiyonunu kullanın.

GetObjectField - grafik fonksiyonu

GetObjectField(), boyutun adını döndürür. **Index**, döndürülmesi gereken boyutu belirten isteğe bağlı bir tamsayıdır.

Şu konumlarda bir grafikte bu işlevi kullanamazsınız: başlık, alt başlık, alt bilgi, referans çizgisi ifadesi.

Object ID kullanarak bir boyutun veya hesaplamanın adını başka bir nesnede referans veremezsiniz.

Söz Dizimi:

```
GetObjectField ([index])
```

Örnek:

```
GetObjectField(1)
```

Örnek: Grafik ifadesi

Bir grafik ifadesinde GetObjectField fonksiyonunun örneklerini gösteren Qlik Sense tablosu.

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

Hesaplama adının döndürülmesini istiyorsanız **GetObjectMeasure** fonksiyonunu kullanın.

GetObjectMeasure - grafik fonksiyonu

GetObjectMeasure(), hesaplamanın adını döndürür. **Index**, döndürülmesi gereken hesaplamayı belirten isteğe bağlı bir tamsayıdır.

Şu konumlarda bir grafikte bu işlevi kullanamazsınız: başlık, alt başlık, alt bilgi, referans çizgisi ifadesi.

Object ID kullanarak bir boyutun veya hesaplamanın adını başka bir nesnede referans veremezsiniz.

Söz Dizimi:

```
GetObjectMeasure ([index])
```

Örnek:

```
GetObjectMeasure(1)
```

Örnek: Grafik ifadesi

Bir grafik ifadesinde GetObjectMeasure fonksiyonunun örneklerini gösteren Qlik Sense tablosu

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)

Boyut adının döndürülmesini istiyorsanız **GetObjectField** fonksiyonunu kullanın.

GetPossibleCount - grafik fonksiyonu

GetPossibleCount(), tanımlanan alanda olası değerlerin sayısını bulmak için kullanılır. Tanımlanan alan seçimler içeriyorsa, seçili (yeşil) alanlar sayılır. Aksi takdirde ilişkili (beyaz) değerler sayılır. .

Seçimleri içeren alanlarda **GetPossibleCount()** fonksiyonu seçili (yeşil) alanların sayısını döndürür.

Dönüş verileri türü: tamsayı

Söz Dizimi:

```
GetPossibleCount (field_name)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
field_name	Ölçülecek veri aralığını içeren alan.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde, biri **First name** adı için ve biri de **Initials** için olmak üzere, farklı filtre bölmelerine yüklenen iki alan kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde **John** seçildiği varsayılırsa. `GetPossibleCount ([Initials])`	1; çünkü Initials içinde seçimle ilişkili 1 değer var (**First name** içinde **John**).
First name içinde **John** seçildiği varsayılırsa. `GetPossibleCount ([First name])`	1; çünkü **First name** içinde **John** olmak üzere 1 seçim var.

Örnekler	Sonuçlar
First name içinde **Peter** seçildiği varsayılırsa. `GetPossibleCount ([Initials])`	2; çünkü Peter değeri **Initials** içinde 2 değer ile ilişkilidir.
First name içinde hiçbir değer seçilmediği varsayılırsa. `GetPossibleCount ([First name])`	5; çünkü seçim yok ve **First name** içinde 5 benzersiz değer var.
First name içinde hiçbir değer seçilmediği varsayılırsa. `GetPossibleCount ([Initials])`	6; çünkü seçim yok ve **Initials** içinde 6 benzersiz değer var.

Örnekte kullanılan veriler:

```
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 - grafik fonksiyonu

GetSelectedCount(), bir alandaki seçili (yeşil) değerlerin sayısını bulur.

Söz Dizimi:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişkenler	Açıklama
field_name	Ölçülecek veri aralığını içeren alan.
include_ excluded	**True()** olarak ayarlanırsa, geçerli anda diğer alanlardaki seçimler tarafından hariç tutulan seçilen değerler sayıma dahil edilir. False veya atlanmış ise, bu değerler dahil edilmez.
state_name	Belirli bir görselleştirme için seçilen alternatif durumun adı. **state_name** bağımsız değişkeni kullanılırsa yalnızca belirtilen durum adıyla ilişkili seçimler hesaba katılır.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde biri **First name** adı için, biri **Initials** için ve biri de **Has cellphone** için olmak üzere, farklı filtre bölmelerine yüklenen üç alan kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
First name içinde **John** seçildiği varsayılırsa. `GetSelectedCount ([First name])`	1; çünkü **First name** içinde bir değer seçilmiştir.
First name içinde **John** seçildiği varsayılırsa. `GetSelectedCount ([Initials])`	0; çünkü **Initials** içinde değer seçilmemiştir.
.**First name** içinde seçim yokken **Initials** içinde tüm değerleri seçin ve sonra da **Has cellphone** içinde **Yes** değerini seçin. `GetSelectedCount ([Initials], True ())`	6. **Initials**MC ve PD içeren seçimlerde **Has cellphone** değeri **No** olarak ayarlanmış olsa da include_excluded bağımsız değişkenin True() olarak ayarlanması nedeniyle sonuç halen 6'dır.

Örnekte kullanılan veriler:

```
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 Dosya fonksiyonları

Dosya fonksiyonları (sadece kod ifadelerinde kullanılabilir) geçerli anda okunan tablo dosyası hakkında bilgi döndürür. Bu fonksiyonlar, tablo dosyaları dışındaki tüm veri kaynakları için NULL sonucunu döndürür (istisna: **ConnectString( )**).

Dosya fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Attribute

Bu kod fonksiyonu, farklı medya dosyalarının meta etiketlerinin değerini metin olarak döndürür. Şu dosya biçimleri desteklenir: MP3, WMA, WMV, PNG ve JPG. **filename** dosyası yoksa, desteklenen bir dosya biçimi değilse veya **attributename** adında bir meta etiketi içermiyorsa, NULL döndürülür.

```
Attribute (filename, attributename)
```

ConnectString

ConnectString() fonksiyonu, ODBC veya OLE DB bağlantıları için etkin veri bağlantısının adını döndürür. Herhangi bir **connect** deyimi yürütülmemişse veya **disconnect** deyiminden sonra yürütülmüşse, fonksiyon boş bir dize döndürür.

```
ConnectString ()
```

FileBaseName

FileBaseName fonksiyonu, şu anda okunan tablo dosyasının adını içeren, yol veya uzantı olmadan, bir dize döndürür.

```
FileBaseName ()
```

FileDir

FileDir fonksiyonu, şu anda okunan tablo dosyasının dizinine giden yolu içeren bir dize döndürür.

```
FileDir ()
```

FileExtension

FileExtension fonksiyonu, şu anda okunan tablo dosyasının uzantısını içeren bir dize döndürür.

```
FileExtension ()
```

FileName

FileName fonksiyonu, şu anda okunan tablo dosyasının adını içeren, yol olmadan ancak uzantıyı içerecek şekilde, bir dize döndürür.

```
FileName ()
```

FilePath

FilePath fonksiyonu, şu anda okunan tablo dosyasının tam yolunu içeren bir dize döndürür.

```
FilePath ()
```

FileSize

FileSize fonksiyonu, filename dosyasının veya filename belirtilmemişse, şu anda okunan tablo dosyasının bayt cinsinden boyutunu içeren bir tamsayı döndürür.

```
FileSize ()
```

FileTime

FileTime fonksiyonu, filename dosyasının son değişikliğinin tarihi ve saati için bir zaman damgası döndürür. Bir filename belirtilmezse, fonksiyon geçerli anda okunan tablo dosyasına başvurur.

```
FileTime ([ filename ])
```

GetFolderPath

GetFolderPath fonksiyonu, Microsoft Windows *SHGetFolderPath* fonksiyonunun değerini döndürür. Bu fonksiyon, giriş olarak Microsoft Windows klasörünün adını alır ve klasörün tam yolunu döndürür.

```
GetFolderPath ()
```

QvdCreateTime

Bu kod fonksiyonu, bir QVD dosyasından XML üst bilgisi zaman damgasını döndürür (varsa); aksi takdirde NULL döndürür.

```
QvdCreateTime (filename)
```

QvdFieldName

Bu kod fonksiyonu, bir QVD dosyasındaki **fieldno** numaralı alanın adını döndürür. Alan yoksa NULL döndürülür.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Bu kod fonksiyonu bir QVD dosyasındaki alanların sayısını döndürür.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Bu kod fonksiyonu bir QVD dosyasında o anda bulunan kayıtların sayısını döndürür.

```
QvdNoOfRecords (filename)
```

QvdTableName

Bu kod fonksiyonu bir QVD dosyasında depolanan tablonun adını döndürür.

```
QvdTableName (filename)
```

Attribute

Bu kod fonksiyonu, farklı medya dosyalarının meta etiketlerinin değerini metin olarak döndürür. Şu dosya biçimleri desteklenir: MP3, WMA, WMV, PNG ve JPG. **filename** dosyası yoksa, desteklenen bir dosya biçimi değilse veya **attributename** adında bir meta etiketi içermiyorsa, NULL döndürülür.

Söz Dizimi:

```
Attribute(filename, attributename)
```

Çok sayıda meta etiketi okunabilir. Bu konudaki örneklerde, desteklenen ilgili dosya türleri için hangi etiketlerin okunabildiği gösterilmektedir.

Yalnızca, uygun teknik özelliğe göre dosyada kayıtlı meta etiketleri okuyabilirsiniz (örneğin, MP3 dosyaları için ID2v3 veya JPG dosyaları için EXIF); ***Windows Dosya Gezgini*** *içinde kayıtlı meta bilgilerini okuyamazsınız.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse, klasör veri bağlantısı olarak yol bilgisini de içeren medya dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data*
attributename	Bir meta etiketinin adı.

Örnekler, medya dosyalarının yollarını bulmak için **GetFolderPath** fonksiyonunu kullanır. **GetFolderPath** yalnızca eski modda desteklendiğinden, bu işlevi standart moddda veya Qlik Sense SaaS ile kullandığınızda **GetFolderPath** referanslarını lib:// veri bağlantısı yoluyla değiştirmeniz gerekir.

Dosya sistemi erişim kısıtlaması (page 824)

Example 1: MP3 dosyaları

Bu kod, *MyMusic* klasöründeki tüm olası MP3 meta etiketlerini okur.

```
// 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

Bu kod, *MyPictures* klasöründeki JPG dosyalarından tüm olası EXIF meta etiketlerini okur.

```
// 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: Windows medya dosyaları

Bu kod, *MyMusic* klasöründeki tüm olası WMA/WMV ASF meta etiketlerini okur.

```
// 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

Bu kod, *MyPictures* klasöründeki tüm olası PNG meta etiketlerini okur.

```
// 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

ConnectString() fonksiyonu, ODBC veya OLE DB bağlantıları için etkin veri bağlantısının adını döndürür. Herhangi bir **connect** deyimi yürütülmemişse veya **disconnect** deyiminden sonra yürütülmüşse, fonksiyon boş bir dize döndürür.

Söz Dizimi:

```
ConnectString()
```

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç

`LIB CONNECT TO 'Tutorial ODBC';`
`ConnectString:`
`Load ConnectString() as ConnectString`
`AutoGenerate 1;`
	ConnectString alanında "Tutorial ODBC" döndürür.
	
	Bu örnekler, Tutorial ODBC adlı kullanılabilir veri bağlantınızın olduğunu varsayar.

FileBaseName

FileBaseName fonksiyonu, şu anda okunan tablo dosyasının adını içeren, yol veya uzantı olmadan, bir dize döndürür.

Söz Dizimi:

```
FileBaseName()
```

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt`	Okunan her kayıttaki X alanında 'abc' sonucunu döndürür.

FileDir

FileDir fonksiyonu, şu anda okunan tablo dosyasının dizinine giden yolu içeren bir dize döndürür.

Söz Dizimi:

```
FileDir()
```

Bu fonksiyon, yalnızca standart modda klasör veri bağlantılarını destekler.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`Load *, filedir( ) as X from C:\UserFiles\abc.txt`	Okunan her kayıttaki X alanında 'C:\UserFiles' sonucunu döndürür.

FileExtension

FileExtension fonksiyonu, şu anda okunan tablo dosyasının uzantısını içeren bir dize döndürür.

Söz Dizimi:

```
FileExtension()
```

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Okunan her kayıttaki X alanında 'txt' sonucunu döndürür.

FileName

FileName fonksiyonu, şu anda okunan tablo dosyasının adını içeren, yol olmadan ancak uzantıyı içerecek şekilde, bir dize döndürür.

Söz Dizimi:

```
FileName()
```

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD *, FileName( ) as X from C:\UserFiles\abc.txt`	Okunan her kayıttaki X alanında 'abc.txt' sonucunu döndürür.

FilePath

FilePath fonksiyonu, şu anda okunan tablo dosyasının tam yolunu içeren bir dize döndürür.

Söz Dizimi:

```
FilePath()
```

Bu fonksiyon, yalnızca standart modda klasör veri bağlantılarını destekler.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Okunan her kayıttaki X alanında 'C:\UserFiles\abc.txt' sonucunu döndürür.

FileSize

FileSize fonksiyonu, filename dosyasının veya filename belirtilmemişse, şu anda okunan tablo dosyasının bayt cinsinden boyutunu içeren bir tamsayı döndürür.

Söz Dizimi:

FileSize([filename]**)**

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web dosyası veri bağlantısı olarak bir yol içeren dosyanın adı. Dosya adı belirtmezseniz o anda okunan tablo dosyası kullanılır. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data* • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD *, FileSize( ) as X from abc.txt;`	Okunan her kayıttaki X alanında, belirtilen dosyanın (abc.txt) boyutunu bir tamsayı olarak döndürür.
`FileSize( 'lib://DataFiles/xyz.xls' )`	xyz.xls dosyasının boyutunu döndürür.

FileTime

FileTime fonksiyonu, filename dosyasının son değişikliğinin tarihi ve saati için bir zaman damgası döndürür. Bir filename belirtilmezse, fonksiyon geçerli anda okunan tablo dosyasına başvurur.

Söz Dizimi:

```
FileTime([ filename ])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web dosyası veri bağlantısı olarak bir yol içeren dosyanın adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data* • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD *, FileTime( ) as X from abc.txt;`	Okunan her kayıttaki X alanında, dosyanın (abc.txt) en son değişikliğinin tarihini ve saatini zaman damgası olarak döndürür.
FileTime('xyz.xls')	xyz.xls dosyasının en son değişikliğinin zaman damgasını döndürür.

GetFolderPath

GetFolderPath fonksiyonu, Microsoft Windows *SHGetFolderPath* fonksiyonunun değerini döndürür. Bu fonksiyon, giriş olarak Microsoft Windows klasörünün adını alır ve klasörün tam yolunu döndürür.

Bu fonksiyon, standart modda desteklenmez.

Söz Dizimi:

```
GetFolderPath(foldername)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
foldername	Microsoft Windows klasörünün adı. Klasör adı boşluk içermemelidir. Windows Explorer içinde görünen klasör adındaki herhangi bir boşluk klasör adından kaldırılmalıdır. Örnekler: *MyMusic* *MyDocuments*

Örnekler ve sonuçlar:

Bu örneğin amacı, aşağıdaki Microsoft Windows klasörlerinin yollarını almaktır: *MyMusic*, *MyPictures* ve *Windows*. Örnek kodu uygulamanıza ekleyin ve yeniden yükleyin.

```
LOAD   GetFolderPath('MyMusic') as MyMusic,  GetFolderPath('MyPictures') as MyPictures,
GetFolderPath('Windows') as Windows AutoGenerate 1;
```

Uygulama yeniden yüklendikten sonra, veri modeline *MyMusic*, *MyPictures* ve *Windows* eklenir. Her alan, girişte tanımlanan klasörün yolunu içerir. Örneğin:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Bu kod fonksiyonu, bir QVD dosyasından XML üst bilgisi zaman damgasını döndürür (varsa); aksi takdirde NULL döndürür.

Söz Dizimi:

```
QvdCreateTime(filename)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web veri bağlantısı olarak bir yol içeren QVD dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data* • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnek:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Bu kod fonksiyonu, bir QVD dosyasındaki **fieldno** numaralı alanın adını döndürür. Alan yoksa NULL döndürülür.

Söz Dizimi:

```
QvdFieldName(filename , fieldno)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web veri bağlantısı olarak bir yol içeren QVD dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:**data*\ • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data*\ • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*
fieldno	QVD dosyasında bulunan tablonun içindeki alanın numarası.

Örnekler:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Üç örnek de QVD dosyasında yer alan tablonun beşinci alanının adını döndürür.

QvdNoOfFields

Bu kod fonksiyonu bir QVD dosyasındaki alanların sayısını döndürür.

Söz Dizimi:

```
QvdNoOfFields(filename)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web veri bağlantısı olarak bir yol içeren QVD dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:**data*\ • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data*\ • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Örnek: Bu kod fonksiyonu bir QVD dosyasında o anda bulunan kayıtların sayısını döndürür.

Söz Dizimi:

```
QvdNoOfRecords(filename)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web veri bağlantısı olarak bir yol içeren QVD dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:**data*\ • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data*\ • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Bu kod fonksiyonu bir QVD dosyasında depolanan tablonun adını döndürür.

Söz Dizimi:

```
QvdTableName(filename)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
filename	Gerekirse klasör veya web veri bağlantısı olarak bir yol içeren QVD dosyasının adı. **Örnek:** *'lib://Table Files/'* Eski kod oluşturma modunda, aşağıdaki yol biçimleri de desteklenir: • mutlak **Örnek:** *c:\data* • Qlik Sense uygulama çalışma dizinine göreceli. **Örnek:** *data* • İnternet veya intranet üzerinde bulunan bir konuma işaret eden URL adresi (HTTP veya FTP). **Örnek:** *http://www.qlik.com*

Örnekler:

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Finansal fonksiyonlar

Finansal fonksiyonlar, ödemeleri ve faiz oranlarını hesaplamak üzere veri kod dosyasında ve grafik ifadelerinde kullanılabilir.
Tüm bağımsız değişkenler için, ödenen nakit negatif sayılarla temsil edilir. Alınan nakit pozitif sayılarla belirtilir.
Burada, finansal fonksiyonlarda (**range**- ile başlayanlar dışında) kullanılan bağımsız değişkenler listelenmektedir.

> *Tüm finansal fonksiyonlarda, **rate** ve **nper** için birimleri belirtirken tutarlı olmanız çok önemlidir. Beş yıllık bir kredi için aylık ödemeler %6 yıllık faizle yapılıyorsa, **rate** için 0,005 (%6/12) ve **nper** için 60 (5*12) kullanın. Aynı kredi için yıllık ödeme yapılıyorsa, **rate** için %6 ve **nper** için 5 kullanın.*

Finansal fonksiyonlara genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

FV

Bu fonksiyon, dönemsel, sabit ödemeler ve basit yıllık faize göre bir yatırımın gelecekteki değerini döndürür.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Bu fonksiyon, dönemsel, sabit ödemeler ve sabit faiz oranına göre bir yatırımın dönem sayısını döndürür.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Bu fonksiyon, dönemsel, sabit ödemeler ve sabit faiz oranına göre bir kredinin ödemesini döndürür. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Bu fonksiyon, bir yatırımın bugünkü değerini döndürür.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Bu fonksiyon, yıllık gelirin her dönemi için faiz oranını verir. Sonucun varsayılan sayı biçimi **Fix** iki ondalık basamak ve % işaretidir.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Black and Scholes modeli, finansal piyasa türevi araçlar için bir matematik modelidir. Formül bir seçeneğin teorik değerini hesaplar. Qlik Sense uygulamasında, **BlackAndSchole** fonksiyonu değerleri Black and Scholes değiştirilmemiş formülüne (Avrupa stili seçenekler) göre döndürür.

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate ,
type)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
strike	Stokun gelecekteki alım fiyatı.
time_left	Kalan dönem sayısı.
underlying_ price	Stokun mevcut değeri.
vol	Zaman dönemine göre ondalık şekilde yüzde olarak ifade edilen dalgalanma değeri (stok fiyatına ait).
risk_free_rate	Zaman dönemine göre ondalık şekilde yüzde olarak ifade edilen risksiz oran.
call_or_put	Seçeneğin türü: Alım opsiyonları için 'c', 'call' veya sıfır olmayan herhangi bir sayısal değer. Satım opsiyonları için 'p', 'put' veya 0.

Sınırlamalar:

strike, time_left ve underlying_price değerleri >0 olmalıdır.

vol ve risk_free_rate değerleri şöyle olmalıdır: <0 veya >0.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` Bu, bugünkü değeri 68,5 olan bir hisse senedini 4 yıl içinde 130 değerinden satın alma opsiyonunun teorik fiyatını hesaplar. Formül yıllık 0,4 (%40) dalgalanma değeri ve 0,04 (%4) risksiz faiz oranı kullanır.	11,245 döndürür

FV

Bu fonksiyon, dönemsel, sabit ödemeler ve basit yıllık faize göre bir yatırımın gelecekteki değerini döndürür.

Söz Dizimi:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Dönüş verileri türü: sayısal. Sonuç, para için varsayılan sayı biçimine sahiptir. .

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
rate	Dönem başına faiz oranı.
nper	Bir yıllık gelirde bulunan ödeme dönemlerinin toplam sayısı.
pmt	Her dönem yapılan ödeme. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.
pv	Bugünkü değer veya gelecekte yapılacak bir dizi ödemenin şu anki değeri olan toplu miktar. **pv** atlanırsa, 0 (sıfır) olduğu kabul edilir.
type	Ödemeler vadesi dönem sonundaysa 0 ve ödemelerin vadesi dönem başındaysa 1 olmalıdır. **type** atlanırsa, 0 olduğu kabul edilir.

Örnekler ve sonuçlar:

Kod örneği

Örnek	Sonuç
Yeni bir ev aleti için aylık 20 $ olmak üzere 36 taksit ödüyorsunuz. Faiz oranı yıllık %6'dır. Fatura her ay sonunda gelir. Son fatura ödendiğinde, yatırılan toplam nedir? `FV(0.005,36,-20)`	$786.72 döndürür

nPer

Bu fonksiyon, dönemsel, sabit ödemeler ve sabit faiz oranına göre bir yatırımın dönem sayısını döndürür.

Söz Dizimi:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
rate	Dönem başına faiz oranı.
nper	Bir yıllık gelirde bulunan ödeme dönemlerinin toplam sayısı.

Bağımsız Değişken	Açıklama
pmt	Her dönem yapılan ödeme. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.
pv	Bugünkü değer veya gelecekte yapılacak bir dizi ödemenin şu anki değeri olan toplu miktar. **pv** atlanırsa, 0 (sıfır) olduğu kabul edilir.
fv	Gelecekteki değer veya son ödeme yapıldıktan sonra elde etmek istediğiniz nakit bakiyesi. **fv** atlanırsa, 0 olduğu kabul edilir.
type	Ödemeler vadesi dönem sonundaysa 0 ve ödemelerin vadesi dönem başındaysa 1 olmalıdır. **type** atlanırsa, 0 olduğu kabul edilir.

Örnekler ve sonuçlar:

Kod örneği

Örnek	Sonuç
Bir ev aletini aylık 20 $ taksitle satmak istiyorsunuz. Faiz oranı yıllık %6'dır. Fatura her ay sonunda gelir. Son fatura ödendikten sonra alınan paranın değerinin 800 $ değerine eşit olması için kaç dönem gerekir? `nPer(0.005,-20,0,800)`	36,56 döndürür

Pmt

Bu fonksiyon, dönemsel, sabit ödemeler ve sabit faiz oranına göre bir kredinin ödemesini döndürür. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Dönüş verileri türü: sayısal. Sonuç, para için varsayılan sayı biçimine sahiptir. .

Kredi süresince ödenen toplam miktarı bulmak için döndürülen **pmt** değerini **nper** ile çarpın.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
rate	Dönem başına faiz oranı.
nper	Bir yıllık gelirde bulunan ödeme dönemlerinin toplam sayısı.
pv	Bugünkü değer veya gelecekte yapılacak bir dizi ödemenin şu anki değeri olan toplu miktar. **pv** atlanırsa, 0 (sıfır) olduğu kabul edilir.

Bağımsız Değişken	Açıklama
fv	Gelecekteki değer veya son ödeme yapıldıktan sonra elde etmek istediğiniz nakit bakiyesi. **fv** atlanırsa, 0 olduğu kabul edilir.
type	Ödemeler vadesi dönem sonundaysa 0 ve ödemelerin vadesi dönem başındaysa 1 olmalıdır. **type** atlanırsa, 0 olduğu kabul edilir.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
Aşağıdaki formül 8 ayda yüzde 10 yıllık oranla ödenmesi gereken 20.000 $ değerindeki kredinin aylık ödemesini döndürür: `Pmt(0.1/12,8,20000)`	-$2,594.66 döndürür
Aynı kredi için, ödeme dönem başına vadeliyse, ödeme şöyle gerçekleşir: `Pmt(0.1/12,8,20000,0,1)`	-$2,573.21 döndürür

PV

Bu fonksiyon, bir yatırımın bugünkü değerini döndürür.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Dönüş verileri türü: sayısal. Sonuç, para için varsayılan sayı biçimine sahiptir. .

Mevcut değer, gelecekte yapılacak bir dizi ödemenin şu anki değeri olan toplu miktardır. Örneğin, borç para alırken, kredi miktarı borç veren için mevcut değerdir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
rate	Dönem başına faiz oranı.
nper	Bir yıllık gelirde bulunan ödeme dönemlerinin toplam sayısı.
pmt	Her dönem yapılan ödeme. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.
fv	Gelecekteki değer veya son ödeme yapıldıktan sonra elde etmek istediğiniz nakit bakiyesi. **fv** atlanırsa, 0 olduğu kabul edilir.
type	Ödemeler vadesi dönem sonundaysa 0 ve ödemelerin vadesi dönem başındaysa 1 olmalıdır. **type** atlanırsa, 0 olduğu kabul edilir.

Örnekler ve sonuçlar:

Kod örneği

Örnek	Sonuç
%7'lik faiz oranı üzerinden beş yıllık bir dönem boyunca her ayın sonunda 100 $ ödemeniz gereken bir borcun bugünkü değeri nedir? `PV(0.07/12,12*5,-100,0,0)`	$5,050.20 döndürür

Rate

Bu fonksiyon, yıllık gelirin her dönemi için faiz oranını verir. Sonucun varsayılan sayı biçimi **Fix** iki ondalık basamak ve % işaretidir.

Söz Dizimi:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Dönüş verileri türü: sayısal.

rate, yinelemeyle hesaplanır ve sıfır veya daha fazla çözümü olabilir. **rate** fonksiyonunun ardışık sonuçları yakınsamıyorsa NULL değer döndürülür.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
nper	Bir yıllık gelirde bulunan ödeme dönemlerinin toplam sayısı.
pmt	Her dönem yapılan ödeme. Yıllık gelirin ömrü süresince bu değiştirilemez. Ödeme bir negatif sayı olarak (örneğin, -20) belirtilir.
pv	Bugünkü değer veya gelecekte yapılacak bir dizi ödemenin şu anki değeri olan toplu miktar. **pv** atlanırsa, 0 (sıfır) olduğu kabul edilir.
fv	Gelecekteki değer veya son ödeme yapıldıktan sonra elde etmek istediğiniz nakit bakiyesi. **fv** atlanırsa, 0 olduğu kabul edilir.
type	Ödemeler vadesi dönem sonundaysa 0 ve ödemelerin vadesi dönem başındaysa 1 olmalıdır. **type** atlanırsa, 0 olduğu kabul edilir.

Örnekler ve sonuçlar:

Kod örneği

Örnek	Sonuç
Aylık 300 $ ödemeli beş yılda ödenecek 10,000 $'lık kredinin faiz oranı nedir? `Rate(60,-300,10000)`	2.00% döndürür

5.12 Biçimlendirme fonksiyonları

Biçimlendirme fonksiyonları, görüntüleme biçimini giriş sayısal alanlarına veya ifadelere zorla kabul ettirir. Veri türüne bağlı olarak ondalık ayırıcı, binlik ayırıcı vs. için karakterleri belirtebilirsiniz.

Fonksiyonların tümü hem dize hem de sayısal değer içeren bir ikili değer döndürür; ancak bu, sayıdan dizeye bir dönüştürme yapılıyormuş gibi düşünülebilir. **Dual()** özel bir durumdur, ancak diğer biçimlendirme fonksiyonları giriş ifadesinin sayısal değerini alır ve sayıyı temsil eden bir dize oluşturur.

Buna karşılık, yorumlama fonksiyonları bunun tersini yapar: dize ifadelerini alıp sayı olarak değerlendirir ve elde edilen sayının biçimini belirtir.

Fonksiyonlar hem veri kod dosyalarında hem de grafik ifadelerinde kullanılabilir.

Tüm sayısal gösterimler, ondalık ayırıcı olarak nokta kullanılarak verilmiştir.

Biçimlendirme fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

ApplyCodepage

ApplyCodepage(), ifadede belirtilen alan veya metne farklı bir kod sayfası karakter kümesi uygular. **codepage** bağımsız değişkeni, sayı biçiminde olmalıdır.

```
ApplyCodepage (text, codepage)
```

Date

Date(), veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimi veya sağlanmışsa bir biçim dizesini kullanarak bir ifadeyi tarih olarak biçimlendirir.

```
Date (number[, format])
```

Dual

Dual() bir sayı ve dizeyi tek bir kayıtta birleştirerek kaydın sayı temsilinin sıralama ve hesaplama, dize değerinin ise görüntüleme amaçları için kullanılmasını sağlar.

```
Dual (text, number)
```

Interval

Interval(), bir sayıyı veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimi veya sağlanmışsa bir biçim dizesini kullanarak zaman aralığı olarak biçimlendirir.

```
Interval (number[, format])
```

Money

Money(), bir biçim deseni, isteğe bağlı ondalık ve binlik ayırıcılar sağlanmadığı sürece, bir ifadeyi veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimde sayısal olarak para değeri şeklinde biçimlendirir.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() bir sayıyı biçimlendirir, yani ikinci parametrede belirtilen biçimi kullanarak metin görüntülemek için girişin sayısal değerini dönüştürür. İkinci parametre atlanırsa, veri yükleme komut dosyasında ayarlanan ondalık ve binlik ayırıcıları kullanır. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time(), bir biçim dizesi sağlanmadığı sürece, bir ifadeyi veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan zaman biçiminde zaman değeri olarak biçimlendirir.

```
Time (number[, format])
```

Timestamp

TimeStamp(), bir biçim dizesi sağlanmadığı sürece, bir ifadeyi veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan zaman damgası biçiminde tarih ve saat değeri olarak biçimlendirir.

```
Timestamp (number[, format])
```

Ayrıca bkz.

Yorumlama fonksiyonları (page 639)

ApplyCodepage

ApplyCodepage(), ifadede belirtilen alan veya metne farklı bir kod sayfası karakter kümesi uygular. **codepage** bağımsız değişkeni, sayı biçiminde olmalıdır.

> *ApplyCodepage grafik ifadelerinde kullanılabiliyor olsa da, daha yaygın şekilde veri yükleme düzenleyicisinde bir komut dosyası fonksiyonu olarak kullanılır. Örneğin, kontrolünüzün dışında farklı karakter kümeleriyle kaydedilmiş olabilecek dosyaları yüklerken, size gereken karakter kümesini temsil eden kod sayfasını uygulayabilirsiniz.*

Söz Dizimi:

```
ApplyCodepage(text, codepage)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	**codepage** bağımsız değişkeni tarafından verilen ve farklı bir kod sayfası uygulamak istediğiniz alan veya metin.
codepage	**text** tarafından verilen alan veya ifadeye uygulanacak kod sayfasını temsil eden sayı.

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`LOAD ApplyCodepage(ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	SQL'den yüklerken kaynakta farklı karakter kümelerinin bir bileşimi olabilir: UTF-8 biçiminden Kiril, İbranice ve diğerleri. Bunların, her bir satır için farklı bir kod sayfayı uygulanarak satır satır yüklenmesi gerekir. **codepage** değeri 1253, Windows Yunanca karakter kümesini, değer 1255 İbraniceyi ve değer 65001 de standart Latince UTF-8 karakterleri temsil eder.

Ayrıca bkz. *Karakter kümesi (page 109)*

Date

Date(), veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimi veya sağlanmışsa bir biçim dizesini kullanarak bir ifadeyi tarih olarak biçimlendirir.

Söz Dizimi:

```
Date(number[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.

Bağımsız Değişken	Açıklama
format	Sonuçta elde edilen dizenin biçimini açıklayan dize. Hiçbir biçim dizesi sağlanmazsa, veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan tarih biçimi kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde varsayılan ayarların şöyle olduğu kabul edilmektedir:

- Tarih ayarı 1: YY-MM-DD
- Tarih ayarı 2: M/D/YY

Örnek:

```
Date( A )
```
burada A=35648

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	97-08-06	8/6/97
Sayı:	35648	35648

Örnek:

```
Date( A, 'YY.MM.DD' )
```
burada A=35648

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	97.08.06	97.08.06
Sayı:	35648	35648

Örnek:

```
Date( A, 'DD.MM.YYYY' )
```
burada A=35648.375

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	06.08.1997	06.08.1997
Sayı:	35648.375	35648.375

Örnek:

```
Date( A, 'YY.MM.DD' )
```
burada A=8/6/97

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	NULL (hiçbir şey)	97.08.06
Sayı:	NULL	35648

Dual

Dual() bir sayı ve dizeyi tek bir kayıtta birleştirerek kaydın sayı temsilinin sıralama ve hesaplama, dize değerinin ise görüntüleme amaçları için kullanılmasını sağlar.

Söz Dizimi:

```
Dual(text, number)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Sayı bağımsız değişkeni ile birlikte kullanılacak dize değeri.
number	Dize bağımsız değişkeninde dize ile birlikte kullanılacak sayı.

Qlik Sense uygulamasında tüm alan değerleri potansiyel olarak ikili değerlerdir. Bir başka ifadeyle, alan değerleri hem sayısal değer hem de metin değeri alabilir. Bunun bir örneği, 40908 sayısal değerine ve '2011-12-31' metin temsiline sahip olabilen bir tarihtir.

Tek bir alana okunan birden fazla veri öğesi farklı dize temsillerine, ancak aynı geçerli sayı temsiline sahip olduğunda, bu veri öğelerinin tümü karşılaşılan ilk dize temsilini paylaşır.

*Filtre bölmelerinde ve benzeri yerlerde gösterilecek bu ilk dize temsilini oluşturmak amacıyla, **dual** fonksiyon genellikle kodun başında (diğer veriler ilgili alana okunmadan önce) kullanılır.*

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Açıklama
Aşağıdaki örnekleri kodunuza ekleyin ve çalıştırın. `Load dual ( NameDay,NumDay ) as DayOfWeek inline [ NameDay,NumDay Monday,0 Tuesday,1 Wednesday,2 Thursday,3 Friday,4 Saturday,5 Sunday,6 ];`	DayOfWeek alanı bir görselleştirmede örneğin boyut olarak kullanılabilir. Haftanın günlerini içeren bir tabloda günler, alfabetik sıra yerine doğru numara sırasına göre otomatik olarak sıralanır.
`Load Dual('Q' & Ceil (Month(Now())/3), Ceil (Month(Now())/3)) as Quarter AutoGenerate 1;`	Bu örnek geçerli çeyreği bulur. **Now()** fonksiyonu yılın ilk üç ayı içinde çalıştırıldığında Q1 olarak görüntülenir, ikinci üç ay için Q2 olur ve bu şekilde devam eder. Bununla birlikte, Quarter alanı sıralamada kullanıldığında sayısal değerine göre davranış sergiler: 1 ila 4.
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Önceki örnekte olduğu gibi, Quarter alanı 'Q1' ila 'Q4' metin değerleriyle oluşturulur ve 1 ila 4 sayısal değerleri atanır. Bunu kod içinde kullanılabilmek için Date değerlerinin yüklenmesi gerekir.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Bu örnek, '2012-W22' biçiminde metin değerleri ile bir YearWeek alanı oluşturur ve aynı zamanda haftanın ilk gününün tarih numarasına karşılık gelen bir sayısal değer atar; örneğin: 41057. Bunu kod içinde kullanılabilmek için Date değerlerinin yüklenmesi gerekir.

Interval

Interval(), bir sayıyı veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimi veya sağlanmışsa bir biçim dizesini kullanarak zaman aralığı olarak biçimlendirir.

Aralıklar saat olarak, gün olarak veya gün, saat, dakika, saniye ve salisenin bileşimi olarak biçimlendirilebilir.

Söz Dizimi:

```
Interval(number[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.
format	Sonuçta elde edilen aralık dizesinin nasıl biçimlendirileceğini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde varsayılan ayarların şöyle olduğu kabul edilmektedir:

- Tarih biçimi ayarı 1: YY-MM-DD
- Tarih biçimi ayarı 2: hh:mm:ss
- Sayı ondalık ayırıcısı:

Sonuçlar tablosu

Örnek	Dize	Sayı
`Interval( A )` burada A=0,375	09:00:00	0.375
`Interval( A )` burada A=1,375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` burada A=1,375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` burada A=97-08-06 09:00:00 and B=96-08-06 00:00:00	365 09:00	365.375

Money

Money(), bir biçim deseni, isteğe bağlı ondalık ve binlik ayırıcılar sağlanmadığı sürece, bir ifadeyi veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan biçimde sayısal olarak para değeri şeklinde biçimlendirir.

Söz Dizimi:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.
format	Sonuçta elde edilen para dizesinin nasıl biçimlendirileceğini açıklayan dize.
dec_sep	Ondalık sayı ayırıcısını belirten dize.
thou_sep	Binlik sayı ayırıcısını belirten dize.

2-4 bağımsız değişkenleri atlanırsa, işletim sisteminde ayarlanmış para birimi biçimi kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde varsayılan ayarların şöyle olduğu kabul edilmektedir:

- MoneyFormat ayarı 1: kr ##0,00, MoneyThousandSep' '
- MoneyFormat ayarı 2: $ #,##0.00, MoneyThousandSep','

Örnek:

```
Money( A )
```
burada A=35648

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	kr 35 648,00	$ 35,648.00
Sayı:	35648.00	35648.00

Örnek:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
burada A=3564800

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	3,564,800 ¥	3,564,800 ¥
Sayı:	3564800	3564800

Num

Num() bir sayıyı biçimlendirir, yani ikinci parametrede belirtilen biçimi kullanarak metin görüntülemek için girişin sayısal değerini dönüştürür. İkinci parametre atlanırsa, veri yükleme komut dosyasında ayarlanan ondalık ve binlik ayırıcıları kullanır. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

Söz Dizimi:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Dönüş verileri türü: dual

Num fonksiyonu hem dize hem de sayı değeri içeren bir ikili değer döndürür. Fonksiyon, giriş ifadesinin sayısal değerini alır ve sayıyı temsil eden bir dize oluşturur.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.
format	Elde edilen dizenin nasıl biçimlendirileceğini belirten dize. Atlanırsa, veri yükleme kod dosyasında ayarlanan ondalık ve binlik ayırıcılar kullanılır.
dec_sep	Ondalık sayı ayırıcısını belirten dize. Atlanırsa, veri kod yükleme dosyasında ayarlanan DecimalSep değişkeninin değeri kullanılır.
thou_sep	Binlik sayı ayırıcısını belirten dize. Atlanırsa, veri yükleme kod dosyasında ayarlanan ThousandSep değişkeninin değeri kullanılır.

Örnek: Grafik ifadesi

Örnek:

Aşağıdaki tablo, alan A 35648.312'ye eşit olduğunda sonuçları gösterir.

Sonuçlar

Bir	Sonuç
Num(A)	35648,312 (komut dosyasındaki ortam değişkenlerine bağlıdır)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Örnek: Yükleme kodu

Yükleme kodu

Kodda binlik ve ondalık ayırıcılar önceden ayarlanmış olsa da bir sayıyı biçimlendirmek için yükleme kodunda *Num* kullanılabilir. Aşağıdaki yükleme kodu belirli binlik ve ondalık ayırıcıları içerir, ancak verileri farklı şekillerde biçimlendirmek için *Num* öğesini kullanır.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

```
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 ];
```

Yükleme kodunda *Num* fonksiyonunun farklı kullanımlarından elde edilen sonuçları gösteren Qlik Sense tablosu. Tablonun dördüncü sütunu, örnek teşkil etmesi amacıyla yanlış biçimlendirme kullanımını içerir.

Biçimlendirme yok	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

Örnek: Yükleme kodu

Yükleme kodu

Bir sayıyı yüzde olarak biçimlendirmek için yükleme kodunda *Num* kullanılabilir.

Veri yükleme düzenleyicisi'nde yeni bir bölüm oluşturun ve sonra örnek kodu ekleyip çalıştırın. Sonra sonucu görmek için en azından sonuçlar sütununda listelenen alanları uygulamanızdaki bir sayfaya ekleyin.

```
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 ];
```

Yüzdeleri biçimlendirmek için yükleme kodunda kullanılmakta olan *Num* fonksiyonunun sonuçlarını gösteren Qlik Sense tablosu.

İndirim	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time(), bir biçim dizesi sağlanmadığı sürece, bir ifadeyi veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan zaman biçiminde zaman değeri olarak biçimlendirir.

Söz Dizimi:

```
Time(number[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.
format	Sonuçta elde edilen zaman dizesinin nasıl biçimlendirileceğini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde varsayılan ayarların şöyle olduğu kabul edilmektedir:

- Saat biçimi ayarı 1: hh:mm:ss
- Saat biçimi ayarı 2: hh.mm.ss

Örnek:

```
Time( A )
```
burada A=0,375

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	09:00:00	09.00.00
Sayı:	0.375	0.375

Örnek:

```
Time( A )
```
burada A=35648,375

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	09:00:00	09.00.00
Sayı:	35648.375	35648.375

Örnek:

```
Time( A, 'hh-mm' )
```
burada A=0,99999

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	23-59	23-59
Sayı:	0.99999	0.99999

Timestamp

TimeStamp(), bir biçim dizesi sağlanmadığı sürece, bir ifadeyi veri yükleme komut dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan zaman damgası biçiminde tarih ve saat değeri olarak biçimlendirir.

Söz Dizimi:

```
Timestamp(number[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
number	Biçimlendirilecek sayı.
format	Sonuçta elde edilen zaman damgası dizesinin nasıl biçimlendirileceğini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde varsayılan ayarların şöyle olduğu kabul edilmektedir:

- TimeStampFormat ayarı 1: YY-MM-DD hh:mm:ss
- TimeStampFormat ayarı 2: M/D/YY hh:mm:ss

Örnek:

```
Timestamp( A )
```
burada A=35648,375

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	97-08-06 09:00:00	8/6/97 09:00:00
Sayı:	35648.375	35648.375

Örnek:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
burada A=35648

Sonuçlar tablosu

Sonuçlar	Ayar 1	Ayar 2
Dize:	1997-08-06 00.00	1997-08-06 00.00
Sayı:	35648	35648

5.13 Genel sayısal fonksiyonlar

Bu genel sayısal fonksiyonlarda, bağımsız değişkenler, **x** değerinin gerçek değerli bir sayı olarak yorumlanması gereken ifadelerdir. Tüm fonksiyonlar, hem veri kod dosyalarında hem de grafik ifadelerinde kullanılabilir.

Genel sayısal fonksiyonlara genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

bitcount

BitCount(), bir ondalık sayının ikili eş değerinde kaç bitin 1 olarak ayarlandığını döndürür. Yani fonksiyon, **integer_number** içinde ayarlı bitlerin sayısını döndürür; burada **integer_number**, imzalı bir 32 bitlik tam sayı olarak yorumlanır.

```
BitCount(integer_number)
```

div

Div(), birinci bağımsız değişkenin ikinci bağımsız değişkene aritmetik bölümünün tamsayı kısmını döndürür. Her iki parametre de gerçek sayı olarak yorumlanır; yani tamsayı olmaları gerekmez.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs(), **x** sayısının mutlak değerini döndürür. Sonuç pozitif bir sayıdır.

```
Fabs (x)
```

fact

Fact(), **x** pozitif tamsayısının faktöriyelini döndürür.

```
Fact (x)
```

frac

Frac(), **x** öğesinin kesir bölümünü döndürür.

```
Frac (x)
```

sign

Sign(), **x** değerinin bir pozitif sayı, 0 veya negatif sayı olma durumuna bağlı olarak 1, 0 veya -1 döndürür.

```
Sign (x)
```

Kombinasyon ve permütasyon fonksiyonları

combin

Combin(), bir **p** öğeleri kümesinden seçilebilecek **q** öğelerinin birleşimlerinin sayısını döndürür. Formülde de görüldüğü gibi: `Combin(p,q) = p! / q!(p-q)!` Öğelerin seçilme sırası önemli değildir.

```
Combin (p, q)
```

permut

Permut(), bir **p** öğeleri kümesinden seçilebilecek **q** öğelerinin permütasyonlarının sayısını döndürür. Formülde de görüldüğü gibi: `Permut(p,q) = (p)! / (p - q)!` Öğelerin seçilme sırası önemlidir.

```
Permut (p, q)
```

Modulo fonksiyonları

fmod

fmod(), ilk bağımsız değişkenin (bölünen) tamsayı bölümünün ikinci bağımsız değişkenle (bölen) bölümünden kalan parçasını döndüren genel mod fonksiyonudur. Sonuç bir gerçek sayıdır. Her iki bağımsız değişken de gerçek sayı olarak yorumlanır; yani tamsayı olmaları gerekmez.

```
Fmod (a, b)
```

mod

Mod(), tamsayı bölümünün olumsuz olmayan kalan kısmını döndüren bir matematik modu fonksiyonudur. İlk bağımsız değişken bölünen ve ikinci bağımsız değişken bölendir. Her iki bağımsız değişken de tamsayı değerleri olmalıdır.

```
Mod (integer_number1, integer_number2)
```

Parite fonksiyonları

even

Even(), **integer_number** öğesinin çift tamsayı ya da sıfır olması durumunda True (-1) döndürür. **integer_number** tek tamsayıysa False (0) döndürür ve **integer_number** bir tamsayı değilse de NULL döndürür.

```
Even (integer_number)
```

odd

Odd(), **integer_number** öğesinin tek tamsayı ya da sıfır olması durumunda True (-1) döndürür. **integer_number** çift tamsayıysa False (0) döndürür ve **integer_number** bir tamsayı değilse de NULL döndürür.

```
Odd (integer_number)
```

Yuvarlama fonksiyonları

ceil

Ceil(), bir sayıyı **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına doğru yukarı yuvarlar.

```
Ceil (x[, step[, offset]])
```

floor

Floor(), bir sayıyı **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına doğru aşağı yuvarlar.

```
Floor (x[, step[, offset]])
```

round

Round(), **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına yukarı veya aşağı doğru yuvarlama sonucunu döndürür.

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount(), bir ondalık sayının ikili eş değerinde kaç bitin 1 olarak ayarlandığını döndürür. Yani fonksiyon, **integer_number** içinde ayarlı bitlerin sayısını döndürür; burada **integer_number**, imzalı bir 32 bitlik tam sayı olarak yorumlanır.

Söz Dizimi:

```
BitCount(integer_number)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`BitCount ( 3 )`	3, ikili 11 olduğundan, bu ifade 2 döndürür
`BitCount ( -1 )`	-1 ikili biçimde 64 bir olduğundan, bu ifade 64 döndürür

Ceil

Ceil(), bir sayıyı **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına doğru yukarı yuvarlar.

Girilen sayıları aşağı yuvarlayan **floor** fonksiyonu ile karşılaştırın.

Söz Dizimi:

```
Ceil(x[, step[, offset]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
x	Giriş sayısı.
step	Aralık artışı. Varsayılan değer 1'dir.
offset	Adım aralığının tabanını tanımlar. Varsayılan değer 0'dir.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Ceil(2.4 )`	3 döndürür Bu örnekte, adımın boyutu 1'dir ve adım aralığının tabanı 0'dır. Aralıklar: ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	5 döndürür
`Ceil(3.88 ,0.1)`	3,9 döndürür Bu örnekte, aralığın boyutu 0,1'dir ve aralığın tabanı 0'dır. Aralıklar: ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	5 döndürür
`Ceil(1.1 ,1)`	2 döndürür
`Ceil(1.1 ,1,0.5)`	1,5 döndürür Bu örnekte, adımın boyutu 1'dir ve kayma 0,5'tir. Bu, adım aralığının 0,5 olduğu ve 0 olmadığı anlamına gelir. Aralıklar: ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...
`Ceil(1.1 ,1,-0.01)`	1,99 döndürür Aralıklar: ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin(), bir **p** öğeleri kümesinden seçilebilecek **q** öğelerinin birleşimlerinin sayısını döndürür. Formülde de görüldüğü gibi: `Combin(p,q) = p! / q!(p-q)!` Öğelerin seçilme sırası önemli değildir.

Söz Dizimi:

```
Combin(p, q)
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

Tamsayı olmayan öğeler kırpılır.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
Toplam 35 piyango sayısından kaç 7 sayı birleşimi seçilebilir? `Combin( 35,7 )`	6.724.520 döndürür

Div

Div(), birinci bağımsız değişkenin ikinci bağımsız değişkene aritmetik bölümünün tamsayı kısmını döndürür. Her iki parametre de gerçek sayı olarak yorumlanır; yani tamsayı olmaları gerekmez.

Söz Dizimi:

```
Div(integer_number1, integer_number2)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Div( 7,2 )`	3 döndürür
`Div( 7.1,2.3 )`	3 döndürür
`Div( 9,3 )`	3 döndürür
`Div( -4,3 )`	-1 döndürür
`Div( 4,-3 )`	-1 döndürür
`Div( -4,-3 )`	1 döndürür

Even

Even(), **integer_number** öğesinin çift tamsayı ya da sıfır olması durumunda True (-1) döndürür. **integer_number** tek tamsayıysa False (0) döndürür ve **integer_number** bir tamsayı değilse de NULL döndürür.

Söz Dizimi:

```
Even(integer_number)
```

Dönüş verileri türü: Boole

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Even( 3 )`	0 döndürür, False
`Even( 2 * 10 )`	-1 döndürür, True
`Even( 3.14 )`	NULL döndürür

Fabs

Fabs(), **x** sayısının mutlak değerini döndürür. Sonuç pozitif bir sayıdır.

Söz Dizimi:

```
fabs(x)
```

Dönüş verileri türü: sayısal

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`fabs( 2.4 )`	2,4 döndürür
`fabs( -3.8 )`	3,8 döndürür

Fact

Fact(), **x** pozitif tamsayısının faktöriyelini döndürür.

Söz Dizimi:

```
Fact(x)
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

x sayısı bir tamsayı değildir ve kesilir. Pozitif olmayan sayılar NULL döndürür.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Fact( 1 )`	1 döndürür
`Fact( 5 )`	120 döndürür (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	NULL döndürür

Floor

Floor(), bir sayıyı **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına doğru aşağı yuvarlar.

Girilen sayıları yukarı yuvarlayan **ceil** fonksiyonu ile karşılaştırın.

Söz Dizimi:

```
Floor(x[, step[, offset]])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
x	Giriş sayısı.
step	Aralık artışı. Varsayılan değer 1'dir.
offset	Adım aralığının tabanını tanımlar. Varsayılan değer 0'dir.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Floor(2.4)`	2 döndürür 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)`	4 döndürür
`Floor(3.88 ,0.1)`	3,8 döndürür Bu örnekte, aralığın boyutu 0,1'dir ve aralığın tabanı 0'dır. Aralıklar: ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...

Örnekler	Sonuçlar
`Floor(3.88 ,5)`	0 döndürür
`Floor(1.1 ,1)`	1 döndürür
`Floor(1.1 ,1,0.5)`	0,5 döndürür Bu örnekte, adımın boyutu 1'dir ve kayma 0,5'tir. Bu, adım aralığının 0,5 olduğu ve 0 olmadığı anlamına gelir. Aralıklar: ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod(), ilk bağımsız değişkenin (bölünen) tamsayı bölümünün ikinci bağımsız değişkenle (bölen) bölümünden kalan parçasını döndüren genel mod fonksiyonudur. Sonuç bir gerçek sayıdır. Her iki bağımsız değişken de gerçek sayı olarak yorumlanır; yani tamsayı olmaları gerekmez.

Söz Dizimi:

```
fmod(a, b)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
a	Bölünen
b	Bölen

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`fmod( 7,2 )`	1 döndürür
`fmod( 7.5,2 )`	1,5 döndürür
`fmod( 9,3 )`	0 döndürür
`fmod( -4,3 )`	-1 döndürür
`fmod( 4,-3 )`	1 döndürür
`fmod( -4,-3 )`	-1 döndürür

Frac

Frac(), **x** öğesinin kesir bölümünü döndürür.

Kesir, `Frac(x ) + Floor(x ) = x` olacak şekilde tanımlanır. Basitçe ifade edilecek olursa bu, pozitif bir sayının kesirli kısmının, (x) sayısı ile kesirli kısımdan önce gelen tam sayı arasındaki fark olduğu anlamına gelir.

Örneğin: 11,43 sayısının kesirli kısmı = 11,43 - 11 = 0,43

Negatif bir sayı için, örneğin, -1,4 için, Floor(-1.4) = -2 olur ve bize şu sonucu verir:

-1,4 sayısının kesirli kısmı = -1,4 - (-2) = -1,4 + 2 = 0,6

Söz Dizimi:

```
Frac(x)
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
x	Kesir getirilecek sayı.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Frac( 11.43 )`	0,43 döndürür
`Frac( -1.4 )`	0,6 döndürür
Bir zaman damgasının sayısal gösteriminden zaman bileşenini ayıklayarak tarihi kaldırın. `Time(Frac(44518.663888889))`	3:56:00 PM döndürür

Mod

Mod(), tamsayı bölümünün olumsuz olmayan kalan kısmını döndüren bir matematik modu fonksiyonudur. İlk bağımsız değişken bölünen ve ikinci bağımsız değişken bölendir. Her iki bağımsız değişken de tamsayı değerleri olmalıdır.

Söz Dizimi:

```
Mod(integer_number1, integer_number2)
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

integer_number2, 0'dan büyük olmalıdır.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Mod( 7,2 )`	1 döndürür
`Mod( 7.5,2 )`	NULL döndürür
`Mod( 9,3 )`	0 döndürür
`Mod( -4,3 )`	2 döndürür
`Mod( 4,-3 )`	NULL döndürür
`Mod( -4,-3 )`	NULL döndürür

Odd

Odd(), **integer_number** öğesinin tek tamsayı ya da sıfır olması durumunda True (-1) döndürür. **integer_number** çift tamsayıysa False (0) döndürür ve **integer_number** bir tamsayı değilse de NULL döndürür.

Söz Dizimi:

```
Odd(integer_number)
```

Dönüş verileri türü: Boole

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Odd( 3 )`	-1 döndürür, True
`Odd( 2 * 10 )`	0 döndürür, False
`Odd( 3.14 )`	NULL döndürür

Permut

Permut(), bir **p** öğeleri kümesinden seçilebilecek **q** öğelerinin permütasyonlarının sayısını döndürür. Formülde de görüldüğü gibi: `Permut(p,q) = (p)! / (p - q)!` Öğelerin seçilme sırası önemlidir.

Söz Dizimi:

```
Permut(p, q)
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

Tamsayı olmayan bağımsız değişkenler kırpılır.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
8 katılımcısı olan bir 100 metre finalinin ardından altın, gümüş ve bronz madalyalar kaç şekilde dağıtılabilir? `Permut( 8,3 )`	336 döndürür

Round

Round(), **offset** sayısı ile kaydırılan **step**'in en yakın çarpanına yukarı veya aşağı doğru yuvarlama sonucunu döndürür.

Yuvarlanacak sayı bir aralığın tam ortasındaysa, yukarı yuvarlanır.

Söz Dizimi:

```
Round(x[, step[, offset]])
```

Dönüş verileri türü: sayısal

> *Kayan nokta sayısını yuvarlıyorsanız, hatalı sonuçlar görebilirsiniz. Bu yuvarlama hataları, kayan nokta sayılarının sınırlı sayıda ikili basamakla ifade edilmesinden kaynaklanmaktadır. Bu nedenle, sonuçlar zaten yuvarlanmış bir sayı kullanılarak hesaplanır. Bu yuvarlama hataları çalışmanızı etkileyecekse, sayıları çarparak yuvarlamadan önce tamsayılara dönüştürün.*

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
x	Giriş sayısı.
step	Aralık artışı. Varsayılan değer 1'dir.
offset	Adım aralığının tabanını tanımlar. Varsayılan değer 0'dir.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Round(3.8 )`	4 döndürür Bu örnekte, adımın boyutu 1'dir ve adım aralığının tabanı 0'dır. Aralıklar: ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	4 döndürür
`Round(2.5 )`	3 döndürür. Bu örnekte, adımın boyutu 1'dir ve adım aralığının tabanı 0'dır. Aralıklar ...0 <= x <1, 1 <= x <2, **2<= x <3**... şeklindedir
`Round(2,4 )`	4 döndürür. 2 sayısı 4'ün adım aralığının tam yarısında olduğundan yukarı yuvarlanır. Bu örnekte, adımın boyutu 4'dir ve adım aralığının tabanı 0'dır. Aralıklar ...**0 <= x <4**, 4 <= x <8, 8<= x <12... şeklindedir
`Round(2,6 )`	0 döndürür. 2 sayısı 6'nın adım aralığının yarısından küçük olduğundan aşağı yuvarlanır. Bu örnekte, adımın boyutu 6'dir ve adım aralığının tabanı 0'dır. Aralıklar ...**0 <= x <6**, 6 <= x <12, 12<= x <18... şeklindedir
`Round(3.88 ,0.1)`	3,9 döndürür Bu örnekte, adımın boyutu 0,1'dir ve adım aralığının tabanı 0'dır. Aralıklar: ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	3,889 döndürür Bu örnekte, adımın boyutu 0,001'dir ve adım büyük tam sayıya yuvarlanarak üç ondalık konumla sınırlandırılır.
`Round(3.88 ,5)`	5 döndürür
`Round(1.1 ,1,0.5)`	1,5 döndürür Bu örnekte, adımın boyutu 1'dir ve adım aralığının tabanı 0,5'dır. Aralıklar: ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign(), **x** değerinin bir pozitif sayı, 0 veya negatif sayı olma durumuna bağlı olarak 1, 0 veya -1 döndürür.

Söz Dizimi:

```
Sign(x)
```

Dönüş verileri türü: sayısal

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

Örnekler ve sonuçlar

Örnekler	Sonuçlar
`Sign( 66 )`	1 döndürür
`Sign( 0 )`	0 döndürür
`Sign( - 234 )`	-1 döndürür

5.14 Jeo-uzamsal fonksiyonlar

Bu fonksiyonlar, harita görselleştirmelerinde jeo-uzamsal verileri yönetmek için kullanılır. Qlik Sense, jeo-uzamsal veriler için GeoJSON belirtimlerini izler ve şunları destekler:

- Point
- Linestring
- Polygon
- Multipolygon

GeoJSON belirtimleri hakkında daha fazla bilgi için bkz:
GeoJSON.org

Jeo-uzamsal fonksiyonlara genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

İki jeo-uzamsal fonksiyon kategorisi vardır: toplama ve toplama olmayan.

Toplama işlevleri, geometri kümesini (noktalar veya alanlar) giriş olarak alır ve tek bir geometri döndürür. Örneğin, birden fazla alan birleştirilebilir ve haritada toplama için tek bir sınır çizilebilir.

Toplama olmayan işlevler tek bir geometri alır ve bir geometri döndürür. Örneğin, GeoGetPolygonCenter() fonksiyonunda, bir alanın sınır geometrisi giriş olarak ayarlanırsa, bu alanın merkezindeki nokta geometrisi (enlem ve boylam) döndürülür.

Aşağıdakiler toplama işlevleridir:

GeoAggrGeometry

GeoAggrGeometry(), birkaç alanı büyük bir alanda birleştirmek için kullanılabilir; örneğin birkaç alt bölge tek bir bölgede birleştirilebilir.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox(), bir geometriyi alanda birleştirmek ve tüm koordinatları içeren en küçük sınırlama kutusunu hesaplamak için kullanılır.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

Bir poligon geometrisinin içerdiği köşe sayısını bulmak için **GeoCountVertex()** kullanılır.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry(), geometriyi bir alanda birleştirmek ve bir projeksiyonun tersini uygulamak için kullanılır.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry(), geometriyi bir alanda birleştirmek ve bir projeksiyon uygulamak için kullanılır.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry(), bir geometrinin köşelerini azaltmak ve her alanın sınır çizgilerini görüntülemeye devam ederek birkaç alanı tek bir alanda birleştirmek için kullanılır.

```
GeoReduceGeometry (geometry)
```

Aşağıdakiler toplama olmayan işlevlerdir:

GeoGetBoundingBox

GeoGetBoundingBox(), bir geometrinin tüm koordinatlarını içeren en küçük jeo-uzamsal sınırlama kutusunu hesaplamak için kodlarda ve grafik ifadelerinde kullanılır.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter(), geometrinin merkez noktasını hesaplamak ve döndürmek için kodlarda ve grafik ifadelerinde kullanılır.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint(), enlem ve boylamla bir nokta oluşturmak ve etiketlemek için kodlarda ve grafik ifadelerinde kullanılır.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject(), bir geometriye projeksiyon uygulamak için kodlarda ve grafik ifadelerinde kullanılır.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry(), birkaç alanı büyük bir alanda birleştirmek için kullanılabilir; örneğin birkaç alt bölge tek bir bölgede birleştirilebilir.

Söz Dizimi:

```
GeoAggrGeometry(field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

Normalde, **GeoAggrGeometry()** jeo-uzamsal sınır verilerini birleştirmek için kullanılabilir. Örneğin, her alan için şehirdeki banliyölerde ve satış gelirlerinde posta kodu alanlarınız olabilir. Satış elemanının bölgesi birkaç posta kodu alanını kapsıyorsa, toplam satışları her alan yerine satış bölgesine göre sunmak ve sonuçları renkle doldurulmuş bir haritada göstermek yararlı olabilir.

GeoAggrGeometry(), tek tek banliyö geometrilerinin toplamını hesaplayabilir ve veri modelinde birleştirilmiş bölge geometrisini oluşturabilir. Bu durumda satış bölgesi sınırları ayarlanırsa, veriler yeniden yüklendiğinde birleştirilen yeni sınırlar ve gelir haritada gösterilir.

GeoAggrGeometry() bir toplama fonksiyonu olduğundan, bunu kodda kullanırsanız **Group by** cümlesi içeren bir **LOAD** deyimi gerekir.

GeoAggrGeometry() kullanılarak oluşturulan haritaların sınır çizgileri, birleştirilen alanların çizgileridir. Önceden toplanmış alanların sınır çizgilerini ayrı ayrı görüntülemek isterseniz GeoReduceGeometry() kullanın.

Örnekler:
Bu örnekte, alan verileri içeren bir KML dosyası yüklenir ve ardından, toplanmış alan verilerini içeren bir tablo yüklenir.

```
[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(), bir geometriyi alanda birleştirmek ve tüm koordinatları içeren en küçük sınırlama kutusunu hesaplamak için kullanılır.

GeoBoundingBox, dört değerli bir liste olarak temsil edilir: sol, sağ, üst, alt.

Söz Dizimi:

```
GeoBoundingBox(field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

GeoBoundingBox(), bir geometri kümesini birleştirir ve birleştirilen geometrinin tüm koordinatlarını içeren en küçük dikdörtgen için dört koordinat döndürür.

Sonucu haritada görselleştirmek için dört koordinattan elde edilen dizeyi poligon biçimine aktarın, aktarılan alanı jeo-poligon biçimiyle etiketleyin ve bu alanı harita nesnesine sürükleyip bırakın. Dikdörtgen kutular harita görselleştirmesinde görüntülenecektir.

GeoCountVertex

Bir poligon geometrisinin içerdiği köşe sayısını bulmak için **GeoCountVertex()** kullanılır.

Söz Dizimi:

```
GeoCountVertex(field_name)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

GeoGetBoundingBox

GeoGetBoundingBox(), bir geometrinin tüm koordinatlarını içeren en küçük jeo-uzamsal sınırlama kutusunu hesaplamak için kodlarda ve grafik ifadelerinde kullanılır.

GeoBoundingBox() fonksiyonu tarafından oluşturulan jeo-uyamsal sınırlama kutusu dört değerli bir liste olarak temsil edilir: sol, sağ, üst, alt.

Söz Dizimi:

```
GeoGetBoundingBox(field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

*Bu ve diğer toplama olmayan jeo-uyamsal fonksiyonlarla veri yükleme düzenleyicisinde **Group by** cümlesini kullanmayın; aksi halde yükleme sırasında hata oluşabilir.*

GeoGetPolygonCenter

GeoGetPolygonCenter(), geometrinin merkez noktasını hesaplamak ve döndürmek için kodlarda ve grafik ifadelerinde kullanılır.

Bazı durumlarda, bir haritada renk dolgusu yerine bir nokta çizmek gerekir. Mevcut jeo-uyamsal veriler yalnızca alan geometrisi biçiminde kullanılabiliyorsa (örneğin, bir sınır), alanın merkezi için enlem ve boylam çifti almak üzere **GeoGetPolygonCenter()** öğesini kullanın.

Söz Dizimi:

```
GeoGetPolygonCenter(field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

*Bu ve diğer toplama olmayan jeo-uyamsal fonksiyonlarla veri yükleme düzenleyicisinde **Group by** cümlesini kullanmayın; aksi halde yükleme sırasında hata oluşabilir.*

GeoInvProjectGeometry

GeoInvProjectGeometry(), geometriyi bir alanda birleştirmek ve bir projeksiyonun tersini uygulamak için kullanılır.

Söz Dizimi:

```
GeoInvProjectGeometry(type, field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
type	Haritanın geometrisinin dönüştürülmesinde kullanılan yansıma türü. Bu, iki değerden birini alabilir: 1:1 yansıma ile sonuçlanan 'birim' (varsayılan) veya standart merkator yansımasını kullanan 'merkator'
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

Örnek:

Kod örneği

Örnek	Sonuç
`Load` deyiminde: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	**AreaPolygon** olarak yüklenen geometri, Merkator yansımasının ters dönüştürmesi kullanılarak dönüştürülür ve görselleştirmelerde kullanılmak üzere **InvProjectGeometry** olarak depolanır.

GeoMakePoint

GeoMakePoint(), enlem ve boylamla bir nokta oluşturmak ve etiketlemek için kodlarda ve grafik ifadelerinde kullanılır. GeoMakePoint, boylam ve enlem sırasıyla noktaları döndürür.

Söz Dizimi:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Dönüş verileri türü: dize, biçimlendirilmiş [enlem, boylam]

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
lat_field_name	Noktanın enlemini temsil eden bir alana referansta bulunan alan veya ifade.
lon_field_name	Noktanın boylamını temsil eden bir alana referansta bulunan alan veya ifade.

*Bu ve diğer toplama olmayan jeo-uyamsal fonksiyonlarla veri yükleme düzenleyicisinde **Group by** cümlesini kullanmayın; aksi halde yükleme sırasında hata oluşabilir.*

GeoProject

GeoProject(), bir geometriye projeksiyon uygulamak için kodlarda ve grafik ifadelerinde kullanılır.

Söz Dizimi:

```
GeoProject(type, field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
type	Haritanın geometrisinin dönüştürülmesinde kullanılan yansıma türü. Bu, iki değerden birini alabilir: 1:1 yansıma ile sonuçlanan 'birim' (varsayılan) veya web merkator yansımasını kullanan 'merkator'.
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

*Bu ve diğer toplama olmayan jeo-uyamsal fonksiyonlarla veri yükleme düzenleyicisinde **Group by** cümlesini kullanmayın; aksi halde yükleme sırasında hata oluşabilir.*

Örnek:

Kod örnekleri

Örnek	Sonuç
`Load` deyiminde: `GeoProject('mercator',Area) as GetProject`	Merkator yansıması, **Area** olarak yüklenen geometriye uygulanır ve sonuç **GetProject** olarak depolanır.

GeoProjectGeometry

GeoProjectGeometry(), geometriyi bir alanda birleştirmek ve bir projeksiyon uygulamak için kullanılır.

Söz Dizimi:

```
GeoProjectGeometry(type, field_name)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
type	Haritanın geometrisinin dönüştürülmesinde kullanılan yansıma türü. Bu, iki değerden birini alabilir: 1:1 yansıma ile sonuçlanan 'birim' (varsayılan) veya web merkator yansımasını kullanan 'merkator'.

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.

Örnek:

Örnek	Sonuç
`Load` deyiminde: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	**AreaPolygon** olarak yüklenen geometri, Merkator yansıması kullanılarak dönüştürülür ve görselleştirmelerde kullanılmak üzere **ProjectGeometry** olarak depolanır.

GeoReduceGeometry

GeoReduceGeometry(), bir geometrinin köşelerini azaltmak ve her alanın sınır çizgilerini görüntülemeye devam ederek birkaç alanı tek bir alanda birleştirmek için kullanılır.

Söz Dizimi:

```
GeoReduceGeometry(field_name[, value])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Temsil edilecek coğrafyayı içeren bir alana başvuran alan veya ifade. Bu, enlem ve boylamı veya bir alanı gösteren bir nokta (veya nokta kümesi) olabilir.
value	Geometriye uygulanacak azaltma miktarı. Aralık 0 ile 1 arasındadır; 0 azaltma yapılmayacağını, 1 ise köşe için maksimum azaltma miktarını gösterir. *Karmaşık veri kümesiyle 0,9 veya daha yüksek value kullanılırsa, köşe sayısı görsel sunumun hatalı olduğu bir değere düşürülebilir.*

GeoReduceGeometry(), birkaç alanı tek bir alana topladığı için **GeoAggrGeometry()** deyimininkine benzer bir fonksiyon gerçekleştirir. Aralarındaki fark, **GeoReduceGeometry()** deyimini kullandığınızda ön toplama verilerindeki sınır çizgilerinin haritada ayrı ayrı gösterilmesidir.

GeoReduceGeometry() bir toplama fonksiyonu olduğundan, bunu kodda kullanırsanız **Group by** cümlesi içeren bir **LOAD** deyimi gerekir.

Örnekler:
Bu örnekte, alan verileri içeren bir KML dosyası yüklenir ve daha sonra azaltılmış ve toplanmış alan verilerini içeren bir tablo yüklenir.

```
[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 Yorumlama fonksiyonları

Yorumlama fonksiyonları, giriş metin alanlarının veya ifadelerinin içeriğini değerlendirir ve sonuçta elde edilen sayısal değere belirtilen bir veri biçimini zorla kabul ettirir. Bu fonksiyonları kullanarak, veri türüne göre ondalık ayırıcı, binlik ayırıcı ve tarih biçimi gibi öznitelikler de dahil olmak üzere sayının biçimini belirtebilirsiniz.

Yorumlama fonksiyonlarının tümü hem dize hem de sayısal değer içeren bir ikili değer döndürür; ancak bu, dizeden sayıya bir dönüştürme yapılıyormuş gibi düşünülebilir. Fonksiyonlar, giriş ifadesinin metin değerini alır ve dizeyi temsil eden bir sayı oluşturur.

Buna karşın, biçimlendirme fonksiyonları tam tersini yapar: Sayısal ifadeleri alıp bunları dize olarak değerlendirir ve sonuçta elde edilen metnin görüntülenme biçimini belirtir.

Herhangi bir yorumlama fonksiyonu kullanılmazsa, Qlik Sense, kod değişkenleri tarafından ve işletim sistemi tarafından tanımlanan varsayılan sayı biçimi, tarih biçimi ve saat biçimi ayarlarını kullanarak verileri sayılar, tarihler, zamanlar, zaman damgaları ve dizelerden oluşan bir karışım olarak yorumlar.

Tüm yorumlama fonksiyonları hem veri kod dosyalarında hem de grafik ifadelerinde kullanılabilir.

Tüm sayısal gösterimler, ondalık ayırıcı olarak nokta kullanılarak verilmiştir.

Yorumlama fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Date#

Date#, bir ifadeyi ikinci bağımsız değişkende (sağlanmışsa) belirtilen biçimde bir tarih olarak değerlendirir. Biçim kodu atlanırsa işletim sisteminde ayarlanmış varsayılan tarih biçimi kullanılır.

```
Date# (page 640)(text[, format])
```

Interval#

Interval#(), bir metin ifadesini, varsayılan olarak işletim sistemindeki ayarlı biçimde veya sağlanmışsa ikinci bağımsız değişkende belirtilen biçimde bir zaman aralığı olarak değerlendirir.

```
Interval# (page 641)(text[, format])
```

Money#

Money#(), bir biçim dizesi sağlanmadığı sürece bir metin dizesini kod dosyasında veya işletim sisteminde ayarlanan biçimde bir para değerine dönüştürür. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

```
Money# (page 642)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#(), bir metin dizesini sayısal bir değer olarak yorumlar, yani ikinci parametrede belirtilen biçimi kullanarak giriş dizesini bir sayıya dönüştürür. İkinci parametre atlanırsa, veri yükleme komut dosyasında ayarlanan ondalık ve binlik ayırıcıları kullanır. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

```
Num# (page 644)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text(), sayısal yorumlama mümkün olsa bile, ifadeyi metin olarak işlem görmeye zorlar.

```
Text(expr)
```

Time#

Time#(), bir ifadeyi bir biçim deseni sağlanmadığı sürece veri kod dosyasında veya işletim sisteminde ayarlanan zaman biçiminde zaman değeri olarak değerlendirir..

```
Time# (page 645)(text[, format])
```

Timestamp#

Timestamp#(), bir ifadeyi biçim deseni sağlanmadığı sürece veri kod dosyasında veya işletim sisteminde ayarlanan zaman damgası biçiminde tarih ve saat değeri olarak değerlendirir.

```
Timestamp# (page 646)(text[, format])
```

Ayrıca bkz.

- *Biçimlendirme fonksiyonları (page 605)*

Date#

Date#, bir ifadeyi ikinci bağımsız değişkende (sağlanmışsa) belirtilen biçimde bir tarih olarak değerlendirir.

Söz Dizimi:

```
Date#(text[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.
format	Değerlendirilecek metin dizesinin biçimini açıklayan dize. Atlandığı takdirde, veri kod dosyasındaki sistem değişkenlerinde veya işletim sisteminde ayarlanan tarih biçimi kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki örnek **M/D/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyasının en üstünde **SET DateFormat** deyimi içinde belirtilir.

Bu örnek kodu uygulamanıza ekleyin ve çalıştırın.

```
Load *,
Num(Date#(StringDate)) as Date;
LOAD * INLINE [
StringDate
8/7/97
8/6/1997
```

Boyutlar olarak **StringDate** ve **Date** ile bir tablo oluştururursanız, sonuçlar şu şekilde olur:

Sonuçlar

StringDate	Tarih
8/7/97	35649
8/6/1997	35648

Interval#

Interval#(), bir metin ifadesini, varsayılan olarak işletim sistemindeki ayarlı biçimde veya sağlanmışsa ikinci bağımsız değişkende belirtilen biçimde bir zaman aralığı olarak değerlendirir.

Söz Dizimi:

```
Interval#(text[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.
format	Dize sayısal aralığa dönüştürülürken kullanılacak beklenen giriş biçimini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır.

interval# fonksiyonu, metin aralığını sayısal bir eşdeğere dönüştürür.

Örnekler ve sonuçlar:
Aşağıdaki örneklerde işletim sistemi ayarlarının şöyle olduğu kabul edilmektedir:

- Kısa tarih biçimi: YY-MM-DD
- Saat biçimi: M/D/YY
- Sayı ondalık ayırıcısı:

Sonuçlar

Örnek	Sonuç
`Interval#( A, 'D hh:mm' )` burada A='1 09:00'	1.375

Money#

Money#(), bir biçim dizesi sağlanmadığı sürece bir metin dizesini kod dosyasında veya işletim sisteminde ayarlanan biçimde bir para değerine dönüştürür. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

Söz Dizimi:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.
format	Dize sayısal aralığa dönüştürülürken kullanılacak beklenen giriş biçimini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlanmış para biçimi kullanılır.
dec_sep	Ondalık sayı ayırıcısını belirten dize. Atlandığı takdirde, veri kod dosyasında ayarlanmış MoneyDecimalSep değeri kullanılır.
thou_sep	Binlik sayı ayırıcısını belirten dize. Atlandığı takdirde, veri kod dosyasında ayarlanmış MoneyThousandSep değeri kullanılır.

money# fonksiyonu genellikle **num#** fonksiyonuyla aynı şekilde davranır; ancak ondalık ayırıcı ve binlik ayıracı için varsayılan değerlerini para biçimine ilişkin kod değişkenlerinden veya para birimine ilişkin sistem ayarlarından alır.

Örnekler ve sonuçlar:
Aşağıdaki örnekler, şu iki işletim sistemi ayarını kabul eder:

- Para biçimi varsayılan ayarı 1: kr # ##0,00
- Para biçimi varsayılan ayarı 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
burada A=35 648,37 kr

Sonuçlar

Sonuçlar	Ayar 1	Ayar 2
Dize	35 648.37 kr	35 648.37 kr
Sayı	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
burada A= $35.648,37

Sonuçlar

Sonuçlar	Ayar 1	Ayar 2
Dize	$35,648.37	$35,648.37
Sayı	35648.37	35648.37

Num#

Num#(), bir metin dizesini sayısal bir değer olarak yorumlar, yani ikinci parametrede belirtilen biçimi kullanarak giriş dizesini bir sayıya dönüştürür. İkinci parametre atlanırsa, veri yükleme komut dosyasında ayarlanan ondalık ve binlik ayırıcıları kullanır. Özel ondalık ve binlik ayırıcı sembolleri isteğe bağlı parametrelerdir.

Söz Dizimi:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Dönüş verileri türü: dual

Num#() fonksiyonu hem dize hem de sayı değeri içeren bir ikili değer döndürür. Fonksiyon, giriş ifadesinin metin gösterimini alır ve bir sayı oluşturur. Sayının biçimini değiştirmez: Çıktı, girişle aynı şekilde biçimlendirilir.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.
format	İlk parametrede kullanılan sayı biçimini belirten dize. Atlanırsa, veri yükleme kod dosyasında ayarlanan ondalık ve binlik ayırıcılar kullanılır.
dec_sep	Ondalık sayı ayırıcısını belirten dize. Atlanırsa, veri kod dosyasında ayarlanan DecimalSep değişkeninin değeri kullanılır.
thou_sep	Binlik sayı ayırıcısını belirten dize. Atlanırsa, veri kod dosyasında ayarlanan ThousandSep değişkeninin değeri kullanılır.

Örnekler ve sonuçlar:
Aşağıdaki tablo, farklı A değerleri için *Num#(A, '#', '.' , ',')* sonucunu gösterir.

Sonuçlar

Bir	Dize temsili	Sayısal değer (burada ondalık nokta ile gösterilir)
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(), sayısal yorumlama mümkün olsa bile, ifadeyi metin olarak işlem görmeye zorlar.

Söz Dizimi:

```
Text (expr)
```

Dönüş verileri türü: dual

Örnek:

```
Text( A )
```
burada A=1234

Sonuçlar

Dize	Sayı
1234	-

Örnek:

```
Text( pi( ) )
```

Sonuçlar

Dize	Sayı
3.1415926535898	-

Time#

Time#(), bir ifadeyi bir biçim deseni sağlanmadığı sürece veri kod dosyasında veya işletim sisteminde ayarlanan zaman biçiminde zaman değeri olarak değerlendirir..

Söz Dizimi:

```
time#(text[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.
format	Değerlendirilecek metin dizesinin biçimini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır.

Örnek:

- Saat biçimi varsayılan ayarı 1: hh:mm:ss
- Saat biçimi varsayılan ayarı 2: hh.mm.ss

```
time#( A )
```
(burada A=09:00:00)

Sonuçlar

Sonuçlar	Ayar 1	Ayar 2
Dize:	09:00:00	09:00:00
Sayı:	0.375	-

Örnek:

- Saat biçimi varsayılan ayarı 1: hh:mm:ss
- Saat biçimi varsayılan ayarı 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
(burada A=09.00)

Sonuçlar

Sonuçlar	Ayar 1	Ayar 2
Dize:	09.00	09.00
Sayı:	0.375	0.375

Timestamp#

Timestamp#(), bir ifadeyi biçim deseni sağlanmadığı sürece veri kod dosyasında veya işletim sisteminde ayarlanan zaman damgası biçiminde tarih ve saat değeri olarak değerlendirir.

Söz Dizimi:

```
timestamp#(text[, format])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Değerlendirilecek metin dizesi.

Bağımsız Değişken	Açıklama
format	Değerlendirilecek metin dizesinin biçimini açıklayan dize. Atlandığı takdirde, işletim sisteminde ayarlı kısa tarih biçimi, saat biçimi ve ondalık ayırıcı kullanılır. Zaman damgaları için ISO 8601 desteklenir.

Örnek:

Aşağıdaki örnek **M/D/YYYY** tarih biçimini kullanır. Tarih biçimi, veri kod dosyasının en üstünde **SET DateFormat** deyimi içinde belirtilir.

Bu örnek kodu uygulamanıza ekleyin ve çalıştırın.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
Dize
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Boyutlar olarak **String** ve **TS** ile bir tablo oluşturursanız, sonuçlar şu şekilde olur:

Sonuçlar

Dize	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 Kayıtlar arası fonksiyonlar

Kayıtlar arası fonksiyonları şu durumlarda kullanılır:

- Geçerli kaydın değerlendirilmesi için verilerin daha önceden yüklenmiş kayıtlarından bir değere ihtiyaç duyulduğu durumlarda, veri kod dosyasında.
- Bir grafiğin veri kümesinden bir başka değere ihtiyaç duyulduğu durumlarda, görselleştirme ifadesinde.

Grafiğin ifadelerinden herhangi birinde grafik kayıtlar arası fonksiyonları kullanıldığında, grafiklerdeki y-değerleri üzerinde sıralamaya veya düz tablolardaki ifade sütunlarına göre sıralamaya izin verilmez. Bu nedenle, söz konusu sıralama alternatifleri otomatik olarak devre dışı bırakılır.

Kendi kendine başvuruda bulunan ifade tanımları yalnızca 100'den az satır içeren tablolarda güvenilir şekilde yapılabilir ancak bu, Qlik altyapısının çalıştırıldığı donanıma bağlı olarak değişiklik gösterebilir.

Satır fonksiyonları

Bu fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir.

Above

Above(), tablodaki bir sütun segmenti dahilinde geçerli satırın üstündeki bir satırda ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar doğrudan üstündeki satırdır. Tablolar dışındaki grafikler için **Above()**, grafiğin düz tablo eşdeğerinde geçerli satırın üstündeki satır için değerlendirme yapar.

```
Above - grafik fonksiyonu([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below(), tablodaki bir sütun segmenti dahilinde geçerli satırın altındaki bir satırda ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar doğrudan altındaki satırdır. Tablolar dışındaki grafikler için **Below()**, grafiğin düz tablo eşdeğerinde geçerli sütunun altındaki satır için değerlendirme yapar.

```
Below - grafik fonksiyonu([TOTAL[<fld{,fld}>]] expression [ , offset [,count
]])
```

Bottom

Bottom(), tablodaki bir sütun segmentinin son (en alt) satırındaki bir ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar en alt satırdır. Tablolar dışındaki grafikler için, grafiğin düz tablo eşdeğerinde geçerli sütunun son satırı üzerinde değerlendirme yapılır.

```
Bottom - grafik fonksiyonu([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top(), bir tabloda sütun segmentinin ilk (en üstteki) satırında bulunan bir ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar en üst satırdır. Tablolar dışındaki grafikler için **Top()** değerlendirmesi, grafiğin düz tablo eşdeğerinde geçerli sütunun ilk satırı üzerinde yapılır.

```
Top - grafik fonksiyonu([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows(), bir tablodaki geçerli sütun segmentinde bulunan satırların sayısını döndürür. **NoOfRows()**, bit eşlem grafikleri için grafiğin düz tablo eşdeğerindeki satır sayısını döndürür.

```
NoOfRows - grafik fonksiyonu([TOTAL])
```

Sütun fonksiyonları

Bu fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir.

Column

Column(), bir düz tabloda **ColumnNo** karşılığı olan sütunda bulunan değeri döndürür (boyutlar göz ardı edilir). Örneğin, **Column(2)** ikinci hesaplama sütununun değerini döndürür.

```
Column - grafik fonksiyonu(ColumnNo)
```

Dimensionality

Dimensionality(), geçerli satır için boyutların sayısını döndürür. Pivot tablolar söz konusu olduğunda fonksiyon, toplama olmayan içeriğe sahip (yani, kısmi toplamlar veya daraltılmış toplamalar içermeyen) boyut sütunlarının toplam sayısını döndürür.

```
Dimensionality - grafik fonksiyonu ( )
```

Secondarydimensionality

SecondaryDimensionality(), toplama olmayan içeriğe sahip (yani, kısmi toplamlar veya daraltılmış toplamalar içermeyen) boyut pivot tablo satırlarının sayısını döndürür. Bu fonksiyon, yatay pivot tablo boyutlarına yönelik **dimensionality()** fonksiyonuyla eşdeğerdir.

```
SecondaryDimensionality - grafik fonksiyonu ( )
```

Alan fonksiyonları

FieldIndex

FieldIndex(), **field_name** alanındaki **value** alan değerinin konumunu döndürür (yükleme sırasına göre).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue(), **field_name** alanının **elem_no** konumunda bulunan değeri döndürür (yükleme sırasına göre).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount(), bir alandaki tekil değerlerin sayısını döndüren bir **tamsayı** fonksiyonudur.

```
FieldValueCount (field_name)
```

Pivot Tablo fonksiyonları

Bu fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir.

After

After(), pivot tablodaki bir satır segmenti içinde bulunan geçerli sütundan sonraki sütunda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür.

```
After - grafik fonksiyonu([TOTAL] expression [ , offset [,n]])
```

Before

Before(), pivot tablodaki bir satır segmenti içinde bulunan geçerli sütundan önceki sütunda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür.

```
Before - grafik fonksiyonu([TOTAL] expression [ , offset [,n]])
```

First

First(), pivot tablodaki geçerli satır segmentinin ilk sütununda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür. Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

```
First - grafik fonksiyonu([TOTAL] expression [ , offset [,n]])
```

Last

Last(), pivot tablodaki geçerli satır segmentinin son sütununda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür. Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

```
Last - grafik fonksiyonu([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo(), bir pivot tablodaki geçerli satır segmentinde bulunan geçerli sütunun sayısını döndürür. İlk sütunun sayısı 1'dir.

```
ColumnNo - grafik fonksiyonu([TOTAL])
```

NoOfColumns

NoOfColumns(), bir pivot tablodaki geçerli satır segmentinde bulunan sütun sayısını döndürür.

```
NoOfColumns - grafik fonksiyonu([TOTAL])
```

Veri kod dosyasında kayıtlar arası fonksiyonları

Exists

Exists(), veri kod dosyasında alana daha önce belirli bir alan değerinin yüklenip yüklenmediğini belirler. Fonksiyon TRUE ya da FALSE sonucunu döndürdüğünden, bir **LOAD** deyiminin veya bir **IF** deyiminin **where** cümlesinde kullanılabilir.

```
Exists (field_name [, expr])
```

LookUp

Lookup() zaten yüklü durumdaki bir tablonun içine bakar ve **match_field_name** alanında **match_field_value** değerinin ilk oluşuna karşılık gelen **field_name** değerini döndürür. Bu tablo, mevcut tablo ya da daha önce yüklenmiş başka bir tablo olabilir.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek(), bir tablodaki bir alanın zaten yüklenmiş bir satırının değerini döndürür. Satır numarası belirtilebilir (tabloda olduğu gibi). Satır numarası belirtilmezse, daha önce yüklenmiş son kayıt kullanılır.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous(), **where** cümlesi nedeniyle atılmamış önceki bir giriş kaydındaki verileri kullanarak **expr** ifadesinin değerini bulur. Bir iç tablonun ilk kaydında, bu fonksiyon NULL sonucunu döndürür.

```
Previous (page 682)(expr)
```

Ayrıca bkz.

- *Aralık fonksiyonları (page 702)*

Above - grafik fonksiyonu

Above(), tablodaki bir sütun segmenti dahilinde geçerli satırın üstündeki bir satırda ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar doğrudan üstündeki satırdır. Tablolar dışındaki grafikler için **Above()**, grafiğin düz tablo eşdeğerinde geçerli satırın üstündeki satır için değerlendirme yapar.

Söz Dizimi:

```
Above([TOTAL] expr [ , offset [,count]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
offset	0'dan büyük bir offsetn belirtildiğinde, ifadenin değerlendirmesi geçerli satırdan n satır daha yukarı taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, Above fonksiyonunun karşılık gelen pozitif offset sayısı ile Below fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** bağımsız değişkeni belirtildiğinde, fonksiyon ilk hücreden yukarı doğru sayarak her **count** tablo satırı için bir adet olmak üzere bir **count** değerleri aralığı döndürür. Bu biçimde, fonksiyon herhangi bir özel aralık fonksiyonuna yönelik bir bağımsız değişken olarak kullanılabilir. *Aralık fonksiyonları (page 702)*
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Bir sütun segmentinin ilk satırında, bunun üzerinde bir satır olmadığından, NULL değeri döndürülür.

Sütun segmenti, geçerli sıralama düzeninde boyutlar için aynı değerlere sahip olan ardışık hücreler kümesi olarak tanımlanır. Kayıtlar arası grafik fonksiyonları, eşdeğer düz tablo grafiğinde en sağdaki boyut hariç tutularak sütun segmentinde hesaplanır. Grafikte yalnızca bir boyut varsa veya TOTAL niteleyicisi belirtilirse, ifade tüm tablo genelinde değerlendirilir.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sınırlamalar:

Yinelemeli çağrılar NULL sonucunu döndürür.

Örnekler ve sonuçlar:

Example 1:

Örnek 1 için tablo görselleştirmesi

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

Bu örnekte gösterilen tablonun ekran görüntüsünde, tablo görselleştirmesi **Customer** boyutundan ve şu hesaplamalardan oluşturulmuştur: `Sum(Sales)` ve `Above(Sum(Sales))`.

Above(Sum(Sales)) sütunu, üzerinde başka bir satır olmaması nedeniyle, **Astrida** öğesini içeren **Customer** satırı için NULL döndürür. **Betacab** satırının sonucu **Astrida** için Sum(Sales) değerini ve **Canutility** sonucu da **Betacab** için **Sum(Sales)** değerini gösterir ve bu böyle devam eder.

Sum(Sales)+Above(Sum(Sales)) etiketli sütun için, **Betacab** satırı **Sum(Sales)** + **Betacab** için **Astrida** değerlerinin toplamından (539+587) elde edilen sonucu gösterir. **Betacab** satırına ilişkin sonuç, **Sum (Sales)** + **Canutility** için **Canutility** değerlerinin toplamından (683+539) elde edilen sonucu gösterir.

`Sum(Sales)+Above(Sum(Sales), 3)` ifadesi kullanılarak oluşturulmuş Above offset 3 etiketli hesaplama, **offset** bağımsız değişkenine (3 olarak ayarlı) sahiptir ve satırdaki değeri geçerli satırdan üç satır yukarı taşıma etkisini oluşturur. Geçerli **Customer** öğesine ilişkin **Sum(Sales)** değerini üç satır yukarıdaki **Customer** öğesinin değerine ekler. İlk üç **Customer** satırı için döndürülen değerler null olur.

Tabloda ayrıca, biri `Sum(Sales)+Above(Sum(Sales))` ifadesinden oluşturulan ve biri de etiketli **Higher?** (`IF (Sum(Sales)>Above(Sum(Sales)), 'Higher')` ifadesinden oluşturulan) olmak üzere daha karmaşık hesaplamalar gösterilmektedir.

Bu fonksiyon tablolar dışında grafiklerde de (örneğin, sütun grafiklerinde) kullanılabilir.

Diğer grafik türleri için grafiği düz tablo eşdeğerine dönüştürerek fonksiyonun ilişkili olduğu satırı kolayca yorumlayabilirsiniz.

Example 2:

Bu örnekte gösterilen tabloların ekran görüntülerinde görselleştirmelere daha çok boyut eklenmiştir: **Month** ve **Product**. Birden fazla boyutu olan grafikler için **Above**, **Below**, **Top** ve **Bottom** fonksiyonlarını içeren ifadelerin sonuçları, sütun boyutlarının Qlik Sense tarafından sıralanma düzenine göre değişir. Qlik Sense, en son sıralanan boyuttan kaynaklanan sütun segmentlerini temel alarak fonksiyonları değerlendirir. Sütun sıralama düzeni, **Sıralama** altındaki özellikler panelinde kontrol edilir ve sütunların tabloda göründüğü düzen olmayabilir.

Örnek 2 için tablo görselleştirmesine ait aşağıdaki ekran görüntüsünde, son sıralanan boyut **Month** olduğundan **Above** fonksiyonu aylara dayalı olarak değerlendirme yapar. Her bir aya (**Jan** ila **Aug**) ilişkin her **Product** değeri için bir dizi sonuç vardır (sütun segmenti). Bunu, bir sonraki **Product** için her bir **Month** değerine ait olmak üzere, sonraki sütun segmentine ilişkin bir seri takip eder. Her bir **Product** öğesine ilişkin her **Customer** değeri için bir sütun segmenti olacaktır.

Örnek 2 için tablo görselleştirmesi

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:

Örnek 3 için tablo görselleştirmesine ait ekran görüntüsünde en son sıralanan boyut **Product** boyutudur. Bu da Product boyutunun, özellikler panelindeki Sıralama sekmesinde 3. konuma taşınmasıyla yapılır. **Above** fonksiyonu her bir **Product** için değerlendirilir ve yalnızca iki ürün bulunduğundan (**AA** ve **BB**), her bir seride tek bir null olmayan sonuç vardır. **Jan** ayına ilişkin **BB** satırında **Above(Sum(Sales))** değeri 46'tır. **AA** satırı için değer null'dur. Herhangi bir ay için her bir **AA** satırındaki değer, AA satırının üstünde başka bir

Product değeri bulunmaması nedeniyle her zaman null çıkar. İkinci seri **AA** ve **BB** satırlarında **Feb** ayı ve **Customer** değeri **Astrida** için değerlendirilir. **Astrida** için tüm aylar değerlendirildiğinde, bu dizi ikinci **Customer**Betacab için tekrarlanır ve bu böyle devam eder.

Örnek 3 için tablo görselleştirmesi

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

Örnek 4

Example 4:	**Sonuç**
Above fonksiyonu, aralık fonksiyonları için giriş olarak kullanılabilir. Örneğin: `RangeAvg (Above(Sum(Sales),1,3))`.	Above() fonksiyonuna ait bağımsız değişkenlerde offset, 1 ve count olarak ayarlıdır. Fonksiyon, sütun segmentinde geçerli satırın hemen üstündeki üç satırda (satır varsa) Sum(Sales) ifadesinin sonuçlarını bulur. Bu üç değer, sağlanan sayı aralığındaki değerlerin ortalamasını bulan RangeAvg() fonksiyonu için giriş olarak kullanılır. Boyut olarak Customer öğesini içeren bir tablo RangeAvg() ifadesi için aşağıdaki sonuçları verir. Astrida - Betacab 587 Canutility 563 Divadip: 603

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *Below - grafik fonksiyonu (page 655)*
- *Bottom - grafik fonksiyonu (page 659)*
- *Top - grafik fonksiyonu (page 684)*
- *RangeAvg (page 705)*

Below - grafik fonksiyonu

Below(), tablodaki bir sütun segmenti dahilinde geçerli satırın altındaki bir satırda ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar doğrudan altındaki satırdır. Tablolar dışındaki grafikler için **Below()**, grafiğin düz tablo eşdeğerinde geçerli sütunun altındaki satır için değerlendirme yapar.

Söz Dizimi:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Bağımsız Değişken	Açıklama
offset	1'den büyük bir **offset**n belirtildiğinde, ifadenin değerlendirmesi geçerli satırdan n satır daha aşağı taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, **Below** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Above** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** parametre belirtildiğinde, fonksiyon ilk hücreden aşağı doğru sayarak her **count** tablo satırı için bir adet olmak üzere bir **count** değerleri aralığı döndürür. Bu biçimde, fonksiyon herhangi bir özel aralık fonksiyonuna yönelik bir bağımsız değişken olarak kullanılabilir. *Aralık fonksiyonları (page 702)*
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Bir sütun segmentinin son satırında, bunun altında bir satır olmadığından, NULL değeri döndürülür.

Sütun segmenti, geçerli sıralama düzeninde boyutlar için aynı değerlere sahip olan ardışık hücreler kümesi olarak tanımlanır. Kayıtlar arası grafik fonksiyonları, eşdeğer düz tablo grafiğinde en sağdaki boyut hariç tutularak sütun segmentinde hesaplanır. Grafikte yalnızca bir boyut varsa veya TOTAL niteleyicisi belirtilirse, ifade tüm tablo genelinde değerlendirilir.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sınırlamalar:

Yinelemeli çağrılar NULL sonucunu döndürür.

Örnekler ve sonuçlar:

Example 1:

Örnek 1 için tablo görselleştirmesi

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	-	-	-	-

Örnek 1 için ekran görüntüsünde gösterilen tabloda, tablo görselleştirmesi **Customer** boyutundan ve şu hesaplamalardan oluşturulmuştur: `Sum(Sales)` ve `Below(Sum(Sales))`.

Below(Sum(Sales)) sütunu, altında başka bir satır olmaması nedeniyle, **Divadip** öğesini içeren **Customer** satırı için NULL döndürür. **Canutility** satırının sonucu **Divadip** için Sum(Sales) değerini ve **Betacab** sonucu da **Canutility** için **Sum(Sales)** değerini gösterir ve bu böyle devam eder.

Tabloda ayrıca, etiketli sütunlarda görebileceğiniz daha karmaşık hesaplamalar gösterilmektedir: `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** ve **Higher?**. Bu ifadeler aşağıdaki paragraflarda açıklandığı gibi çalışır.

Sum(Sales)+Below(Sum(Sales)) etiketli sütun için, **Astrida** satırı **Sum(Sales)** + **Betacab** için **Astrida** değerlerinin toplamından (539+587) elde edilen sonucu gösterir. **Betacab** satırına ilişkin sonuç, **Sum (Sales)** + **Canutility** için **Betacab** değerlerinin toplamından (539+683) elde edilen sonucu gösterir.

`Sum(Sales)+Below(Sum(Sales), 3)` ifadesi kullanılarak oluşturulmuş **Below +Offset 3** etiketli hesaplama, **offset** bağımsız değişkenine (3 olarak ayarlı) sahiptir ve satırdaki değeri geçerli satırdan üç satır aşağı taşıma etkisini oluşturur. Geçerli **Customer** öğesine ilişkin **Sum(Sales)** değerini üç satır aşağıdaki **Customer** öğesinden elde edilen değere ekler. En alt üç **Customer** satırı için döndürülen değerler null olur.

Higher? etiketli hesaplama `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')` ifadesinden oluşturulur. Bu ifade, **Sum(Sales)** hesaplamasında geçerli satırın değerlerini alttaki satır ile karşılaştırır. Geçerli satır daha büyük bir değere sahipse, "Higher" metni çıkış olarak verilir.

Bu fonksiyon tablolar dışında grafiklerde de (örneğin, sütun grafiklerinde) kullanılabilir.

Diğer grafik türleri için grafiği düz tablo eşdeğerine dönüştürerek fonksiyonun ilişkili olduğu satırı kolayca yorumlayabilirsiniz.

Birden fazla boyutu olan grafikler için **Above**, **Below**, **Top** ve **Bottom** fonksiyonlarını içeren ifadelerin sonuçları, sütun boyutlarının Qlik Sense tarafından sıralanma düzenine göre değişir. Qlik Sense, en son sıralanan boyuttan kaynaklanan sütun segmentlerini temel alarak fonksiyonları değerlendirir. Sütun sıralama düzeni, **Sıralama** altındaki özellikler panelinde kontrol edilir ve sütunların tabloda göründüğü düzen olmayabilir.Daha fazla ayrıntı için lütfen **Above** fonksiyonundaki 2. örneğe bakın.

2. Örnek

Example 2:	Sonuç
Below fonksiyonu, aralık fonksiyonları için giriş olarak kullanılabilir. Örneğin: `RangeAvg (Below (Sum(Sales),1,3))`.	**Below()** fonksiyonuna ait bağımsız değişkenlerde offset, 1 ve count olarak ayarlıdır. Fonksiyon, sütun segmentinde geçerli satırın hemen altındaki üç satırda (satır varsa) **Sum(Sales)** ifadesinin sonuçlarını bulur. Bu üç değer, sağlanan sayı aralığındaki değerlerin ortalamasını bulan RangeAvg() fonksiyonu için giriş olarak kullanılır. Boyut olarak **Customer** öğesini içeren bir tablo RangeAvg() ifadesi için aşağıdaki sonuçları verir.
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

- *Above - grafik fonksiyonu (page 651)*
- *Bottom - grafik fonksiyonu (page 659)*
- *Top - grafik fonksiyonu (page 684)*
- *RangeAvg (page 705)*

Bottom - grafik fonksiyonu

Bottom(), tablodaki bir sütun segmentinin son (en alt) satırındaki bir ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar en alt satırdır. Tablolar dışındaki grafikler için, grafiğin düz tablo eşdeğerinde geçerli sütunun son satırı üzerinde değerlendirme yapılır.

Söz Dizimi:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
offset	1'den büyük bir **offset**n belirtildiğinde, ifadenin değerlendirmesi alt satırın üstünde n satır yukarı taşınır. Negatif offset sayısı belirtilmesi, **Bottom** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Top** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir parametre **count** değeri belirtildiğinde, fonksiyon, bir değer yerine, geçerli sütun segmentinin her son **count** satırı için bir adet olmak üzere bir **count** değerleri aralığı döndürür. Bu biçimde, fonksiyon herhangi bir özel aralık fonksiyonuna yönelik bir bağımsız değişken olarak kullanılabilir. *Aralık fonksiyonları (page 702)*
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Sütun segmenti, geçerli sıralama düzeninde boyutlar için aynı değerlere sahip olan ardışık hücreler kümesi olarak tanımlanır. Kayıtlar arası grafik fonksiyonları, eşdeğer düz tablo grafiğinde en sağdaki boyut hariç tutularak sütun segmentinde hesaplanır. Grafikte yalnızca bir boyut varsa veya TOTAL niteleyicisi belirtilirse, ifade tüm tablo genelinde değerlendirilir.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sınırlamalar:

Yinelemeli çağrılar NULL sonucunu döndürür.

Örnekler ve sonuçlar:

Örnek 1 için tablo görselleştirmesi

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

Bu örnekte gösterilen tablonun ekran görüntüsünde, tablo görselleştirmesi **Customer** boyutundan ve şu hesaplamalardan oluşturulmuştur: `Sum(Sales)` ve `Bottom(Sum(Sales))`.

Bottom(Sum(Sales)) sütunu tüm satırlar için 757 döndürür; çünkü alt satırın değeri budur: **Divadip**.

Tabloda ayrıca, biri `Sum(Sales)+Bottom(Sum(Sales))` ifadesinden oluşturulan ve biri de **Bottom offset 3** etiketli (`Sum(Sales)+Bottom(Sum(Sales), 3)` ifadesi kullanılarak oluşturulmuş ve **offset** bağımsız değişkeni 3 olarak ayarlanmış) olmak üzere daha karmaşık hesaplamalar gösterilmektedir. Geçerli satıra ilişkin **Sum (Sales)** değerini alt satırdan itibaren üçüncü satırdan gelen değere ekler (yani, geçerli satır artı **Betacab** değeri).

Örnek: 2

Bu örnekte gösterilen tabloların ekran görüntülerinde görselleştirmelere daha çok boyut eklenmiştir: **Month** ve **Product**. Birden fazla boyutu olan grafikler için **Above**, **Below**, **Top** ve **Bottom** fonksiyonlarını içeren ifadelerin sonuçları, sütun boyutlarının Qlik Sense tarafından sıralanma düzenine göre değişir. Qlik Sense, en son sıralanan boyuttan kaynaklanan sütun segmentlerini temel alarak fonksiyonları değerlendirir. Sütun sıralama düzeni, **Sıralama** altındaki özellikler panelinde kontrol edilir ve sütunların tabloda göründüğü düzen olmayabilir.

İlk tabloda ifade **Month** esas alınarak değerlendirilir ve ikinci tabloda ise **Product** esas alınarak değerlendirilir. **End value** hesaplaması `Bottom(Sum(Sales))` ifadesini içerir. **Month** için alt satır Dec olur ve Dec değeri ve ekran görüntüsünde gösterilen **Product** değeri 22 olur. (Yerden kazanmak için bazı satırlar ekran görüntüsünün dışında düzenlenmiştir.)

Örnek 2 için birinci tablo. Month (Dec) esas alındığında End value hesaplaması için Bottom değeri.

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

Örnek 2 için ikinci tablo. Product (Astrida için BB) esas alındığında End value hesaplaması için Bottom değeri.

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

Daha fazla ayrıntı için lütfen **Above** fonksiyonundaki 2. örneğe bakın.

Örnek 3

Örnek: 3	Sonuç
Bottom fonksiyonu, aralık fonksiyonları için giriş olarak kullanılabilir. Örneğin: `RangeAvg (Bottom (Sum(Sales),1,3))`.	**Bottom()** fonksiyonuna ait bağımsız değişkenlerde offset, 1 ve count olarak ayarlıdır. Fonksiyon, sütun segmentinde alt satırın üstündeki satırdan başlayarak (çünkü offset=1) üç satırda ve bunun üstündeki iki satırda (satır varsa) **Sum(Sales)** ifadesinin sonuçlarını bulur. Bu üç değer, sağlanan sayı aralığındaki değerlerin ortalamasını bulan RangeAvg() fonksiyonu için giriş olarak kullanılır. Boyut olarak **Customer** öğesini içeren bir tablo RangeAvg() ifadesi için aşağıdaki sonuçları verir.
	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 '|');
```

Ayrıca bkz.

Top - grafik fonksiyonu (page 684)

Column - grafik fonksiyonu

Column(), bir düz tabloda **ColumnNo** karşılığı olan sütunda bulunan değeri döndürür (boyutlar göz ardı edilir). Örneğin, **Column(2)** ikinci hesaplama sütununun değerini döndürür.

Söz Dizimi:

```
Column(ColumnNo)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
ColumnNo	Hesaplama içeren tablodaki bir sütunun sütun numarası. *Column() fonksiyonu boyut sütunlarını göz ardı eder.*

Sınırlamalar:

ColumnNo, hesaplaması bulunmayan bir sütuna referansta bulunuyorsa, NULL değeri döndürülür.

Yinelemeli çağrılar NULL sonucunu döndürür.

Örnekler ve sonuçlar:

Örnek: Toplam satışların yüzdesi

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

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
C	CC	19	-	0	505	0.00

Örnek: Seçili müşteri için satışların yüzdesi

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

Örnekler ve sonuçlar

Örnekler	Sonuçlar
Order Value şu ifadeyle bir hesaplama olarak tabloya eklenir: `Sum (UnitPrice*UnitSales)`. Total Sales Value şu ifadeyle bir hesaplama olarak eklenir: `Sum(TOTAL UnitPrice*UnitSales)` % Sales şu ifadeyle bir hesaplama olarak eklenir: `100*Column(1)/Column(2)`	Column(1) sonucu, ilk hesaplama sütunu olması nedeniyle Order Value sütunundan alınır. Column(2) sonucu, ikinci hesaplama sütunu olması nedeniyle Total Sales Value sütunundan alınır. *Toplam satışların yüzdesi (page 663)* örneğinde % Sales sütunundaki sonuçlara bakın.
Customer A seçimini yapın.	Seçim, Total Sales Value ve dolayısıyla da %Sales değerlerini değiştirir. *Seçili müşteri için satışların yüzdesi (page 664)* örneğine bakın.

Örneklerde kullanılan veriler:

```
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 - grafik fonksiyonu

Dimensionality(), geçerli satır için boyutların sayısını döndürür. Pivot tablolar söz konusu olduğunda fonksiyon, toplama olmayan içeriğe sahip (yani, kısmi toplamlar veya daraltılmış toplamalar içermeyen) boyut sütunlarının toplam sayısını döndürür.

Söz Dizimi:

```
Dimensionality ( )
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

Bu fonksiyon yalnızca grafiklerde kullanılabilir. Pivot tablo dışındaki tüm grafik türleri için toplam dışındaki tüm satırlarda bulunan boyut sayısını döndürür ve bu değer 0 olur.

Örnek: Dimensionality kullanan grafik ifadesi

Örnek: Grafik ifadesi
Dimensionality() işlevi; toplanmamış verileri olan bir satırdaki boyut sayısına göre farklı hücre formatlaması kullanmak istediğiniz durumlarda bir pivot tablo ile bir grafik ifadesi olarak kullanılabilir. Bu örnek, belirli bir koşulla eşleşen tablo hücrelerine bir arka plan rengi uygulamak için Dimensionality() işlevini kullanmaktadır.

Komut dosyası

Aşağıdaki grafik ifadesi örneğini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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 ',');
```

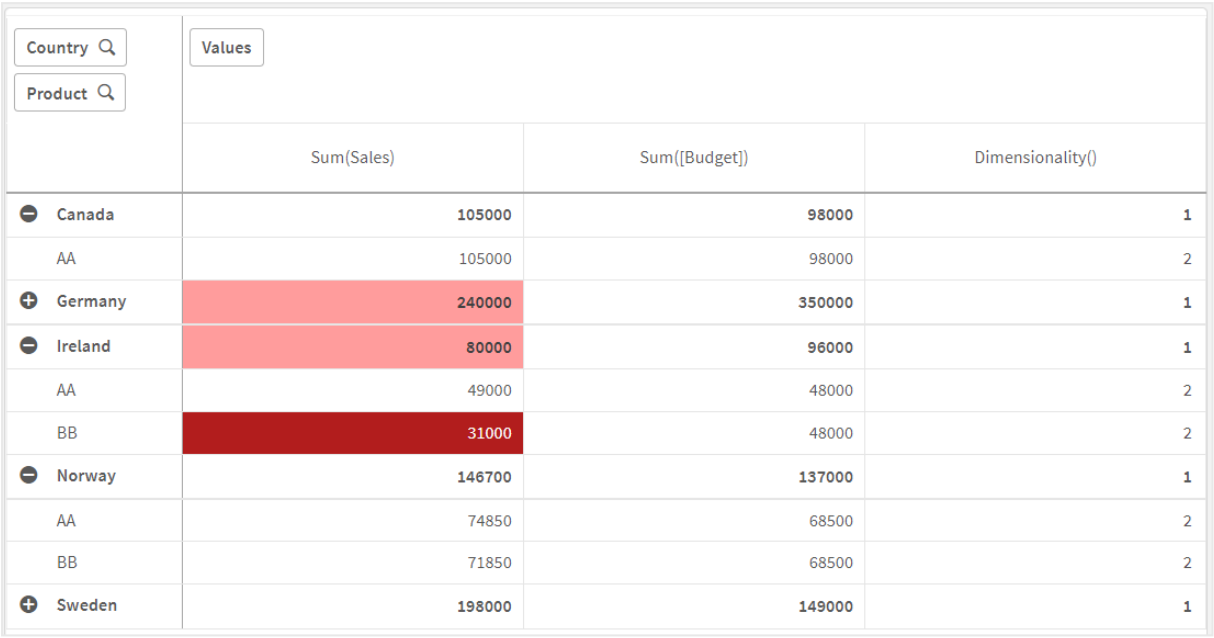
Grafik ifadesi

Bir Qlik Sense sayfasında boyut olarak **Country** ve **Product** ile bir pivot tablo görselleştirmesi oluşturun. Hesaplamalar olarak **Sum(Sales)**, **Sum(Budget)** ve **Dimensionality()** ekleyin.

Özellikler panelinde, **Sum(Sales)** hesaplamasının **Arka plan renk ifadesi** olarak şu ifadeyi girin:

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and
Sum(Sales)<Sum(Budget),RGB(178,29,29) ))
```

Sonuç:



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

Açıklama

`If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` ifadesi, her ürün için Dimensionality değerini ve Sum(Sales) ve Sum(Budget) işlevlerini kontrol eden koşullu deyimler içerir. Koşullar yerine getirilirse, Sum(Sales) değerine bir arka plan rengi uygulanır.

Exists

Exists(), veri kod dosyasında alana daha önce belirli bir alan değerinin yüklenip yüklenmediğini belirler. Fonksiyon TRUE ya da FALSE sonucunu döndürdüğünden, bir **LOAD** deyiminin veya bir **IF** deyiminin **where** cümlesinde kullanılabilir.

*Bir alan değerinin yüklenip yüklenmediğini belirlemek için **Not Exists()** işlevini de kullanabilirsiniz, ancak **Not Exists()** işlevini where cümlesinde kullanıyorsanız dikkatli olmanız önerilir. **Exists()** işlevi hem önceden yüklenen tabloları hem de geçerli tablodaki önceden yüklenen değerleri test eder. Bu nedenle yalnızca ilk oluşum yüklenir. İkinci oluşumla karşılaşıldığında değer zaten yüklenmiştir. Daha fazla bilgi için örneklere bakın.*

Söz Dizimi:

```
Exists(field_name [, expr])
```

Dönüş verileri türü: Boole

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Değer aramak istediğiniz alanın adı. Tırnak işaretleri olmadan belirtik bir alan adı kullanabilirsiniz. Alan, komut dosyası tarafından önceden yüklenmiş olmalıdır. Başka bir deyişle, komut dosyasının daha derinindeki bir cümlede yüklenen bir alanı referans alamazsınız.
expr	Varsa, denetlemek istediğiniz değer. Geçerli load deyiminde bir veya birkaç alanı referans alan bir ifade veya belirtik bir değer kullanabilirsiniz. *Geçerli load deyiminde yer almayan alanları referans alamazsınız.* Bu bağımsız değişken isteğe bağlıdır. Bunu atarsanız işlev, geçerli kayıttaki **field_name** değerinin önceden var olup olmadığını denetler.

Örnekler ve sonuçlar:

1. Örnek

```
Exists (Employee)
```

Geçerli kayıttaki **Employee** alanının değeri bu alanı içeren daha önce okunmuş herhangi bir kayıtta zaten mevcutsa -1 (True) sonucunu döndürür.

`Exists (Employee, Employee)` ve `Exists (Employee)` deyimleri eşdeğerdir.

2. Örnek

```
Exists(Employee, 'Bill')
```

Employee alanının geçerli içeriğinde **'Bill'** alan değeri bulunursa -1 (True) sonucunu döndürür.

Örnek 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;
```

Bu da, Employee ve Address boyutlarını kullanarak tablo görselleştirmesinde kullanabileceğiniz bir tabloyla sonuçlanır.

where yan tümcesi; `where Exists (Employee)`, Citizens tablosundan gelen adlardan yalnızca Employees içinde de bulunan adların yeni tabloya yüklenmesi anlamını taşır. Drop deyimi, karışıklığı önlemek için Employees tablosunu kaldırır.

Sonuçlar

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Örnek 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;
```

where cümlesi, not: `where not Exists (Employee)` ifadesini içerir.

Başka bir deyişle, yalnızca Employees içinde olmayan Citizens tablosundaki adlar yeni tabloya yüklenir.

Citizens tablosunda Lucy için iki değer olduğunu, ancak sonuç tablosunda yalnızca bir değer yer aldığını unutmayın. İlk satırı Lucy değeriyle yüklerseniz, Employee alanına dahil edilir. Bu nedenle ikinci satır kontrol edildiğinde değer zaten mevcuttur.

Sonuçlar

Employee	Address
Mary	London
Lucy	Madrid

Örnek 5

Bu örnekte tüm değerlerin yüklenmesi gösterilmektedir.

```
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;
```

Tüm Lucy değerlerini alabilmek için iki şey değiştirilmiştir:

- Employees tablosundan öncesine bir yükleme eklenmiş; Employee Name olarak yeniden adlandırılmıştır.
  `Load Employee As Name;`
- Citizens içindeki Where koşulu şu şekilde değiştirilmiştir:
  `not Exists (Name, Employee).`

Bu, Name ve Employee için alanlar oluşturur. Lucy olan ikinci satır işaretlendiğinde, Name içinde hala mevcut değildir.

Sonuçlar

Employee	Address
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex(), **field_name** alanındaki **value** alan değerinin konumunu döndürür (yükleme sırasına göre).

Söz Dizimi:

```
FieldIndex(field_name , value)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Endeksin gerekli olduğu alanın adı. Örneğin, sütun bir tablodur. Bir dize değeri olarak verilmelidir. Bu da alan adının tek tırnak içine alınması gerektiği anlamına gelir.
value	**field_name** alanının değeri.

Sınırlamalar:

value öğesi **field_name** alanının alan değerleri arasında bulunamazsa, 0 döndürülür.

Örnekler ve sonuçlar:

Aşağıdaki örnekler alanı kullanır: **Names** tablosundan **First name**.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
Örnek verileri uygulamanıza ekleyin ve çalıştırın.	Örnek verilerde olduğu gibi, **Names** tablosu yüklenir.
Grafik fonksiyonu: First name boyutunu içeren bir tabloda hesaplama olarak ekleyin:	
`FieldIndex ('First name','John')`	1; çünkü 'John' öğesi **First name** alanının yükleme sırasında ilk sırada görünür. Bir filtre bölmesinde **John** öğesinin, yükleme sırasında olduğu gibi değil de alfabetik olarak sıralanması nedeniyle üstten 2. olarak görüneceğini unutmayın.
`FieldIndex ('First name','Peter')`	4; çünkü **FieldIndex()** tek bir değer döndürür; yani yükleme sırasındaki ilk oluşu döndürür.
Kod fonksiyonu: Örnek verilerde olduğu gibi, **Names** tablosu yüklenir:	

`John1:`
 	
`Load FieldIndex('First name','John') as MyJohnPos`
 	
`Resident Names;`
	`MyJohnPos=`1; çünkü 'John' öğesi **First name** alanının yükleme sırasında ilk sırada görünür. Bir filtre bölmesinde **John** öğesinin, yükleme sırasında olduğu gibi değil de alfabetik olarak sıralanması nedeniyle üstten 2. olarak görüneceğini unutmayın.

`Peter1:`
 	
`Load FieldIndex('First name','Peter') as MyPeterPos`
 	
`Resident Names;`
	`MyPeterPos=`4; çünkü **FieldIndex()** tek bir değer döndürür; yani yükleme sırasındaki ilk oluşu döndürür.

Örnekte kullanılan veriler:

```
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(), **field_name** alanının **elem_no** konumunda bulunan değeri döndürür (yükleme sırasına göre).

Söz Dizimi:

```
FieldValue(field_name , elem_no)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Değerin gerekli olduğu alanın adı. Örneğin, sütun bir tablodur. Bir dize değeri olarak verilmelidir. Bu da alan adının tek tırnak içine alınması gerektiği anlamına gelir.
elem_no	Yükleme sırası izlenerek değerin döndürüldüğü alanın konum (öğe) numarası. Bu, tablodaki bir satıra karşılık gelebilir; ancak öğelerin (satırlar) yüklendiği sıraya bağlıdır.

Sınırlamalar:

elem_no, alan değerlerinin sayısından büyükse NULL döndürülür.

Örnek

Komut dosyası

Aşağıdaki kod örneğini oluşturmak için veri yükleme düzenleyicisinde aşağıdaki verileri satır içi yükleme olarak yükleyin.

```
Names:                                  LOAD * inline [ First name|Last name|Initials|Has cellphone John|And
Sue|Brown|SB|Yes Mark|Carr|MC |No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                                   John1:                                                  Load
Names;  Peter1:  Load FieldValue('First name',5) as MyPos2 Resident Names;
```

Görselleştirme oluşturma

Bir Qlik Sense sayfasında bir tablo görselleştirmesi oluşturun. Tabloya **First name**, **MyPos1** ve **MyPos2** alanlarını ekleyin.

Sonuç

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Açıklama

FieldValue('First name','1'), tüm adlar için **MyPos1** değeri olarak John sonucunu döndürür çünkü **First name** alanının yükleme sırasında ilk John görünmektedir. John öğesinin bir filtre bölmesinde, yükleme sırasında olduğu gibi değil de alfabetik olarak sıralanması nedeniyle üstten 2. olarak Jane öğesinden sonra görüneceğini unutmayın.

FieldValue('First name','5'), tüm adlar için **MyPos2** değeri olarak Jane sonucunu döndürür çünkü **First name** alanının yükleme sırasında beşinci olarak Jane görünmektedir.

FieldValueCount

FieldValueCount(), bir alandaki tekil değerlerin sayısını döndüren bir **tamsayı** fonksiyonudur.

Kısmi bir yeniden yükleme veriden değerler kaldırabilir ve bu döndürülen sayıya yansımaz. Döndürülen sayı ilk yeniden yüklemede veya varsa sonraki kısmi bir yeniden yüklemede yüklenen tüm benzersiz değerlere karşılık gelir.

Söz Dizimi:

```
FieldValueCount(field_name)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Değerin gerekli olduğu alanın adı. Örneğin, sütun bir tablodur. Bir dize değeri olarak verilmelidir. Bu da alan adının tek tırnak içine alınması gerektiği anlamına gelir.

Örnekler ve sonuçlar:

Aşağıdaki örneklerde **Names** tablosundaki **First name** alanı kullanılmaktadır.

Örnekler ve sonuçlar

Örnekler	Sonuçlar
Örnek verileri uygulamanıza ekleyin ve çalıştırın.	Örnek verilerde olduğu gibi, **Names** tablosu yüklenir.
Grafik fonksiyonu: First name boyutunu içeren bir tabloda hesaplama olarak ekleyin:	
`FieldValueCount('First name')`	5; çünkü **Peter** iki kez görünür.
`FieldValueCount('Initials')`	6; çünkü **Initials** yalnızca tekil değerlere sahip.

Örnekler	Sonuçlar
Kod fonksiyonu: Örnek verilerde olduğu gibi, **Names** tablosu yüklenir:	

`FieldCount1:`
 	
`Load FieldValueCount('First name') as MyFieldCount1`
 	
`Resident Names;`
	`MyFieldCount1`=5; çünkü 'Peter' iki kez görünür.

`FieldCount2:`
 	
`Load FieldValueCount('Initials') as MyInitialsCount1`
 	
`Resident Names;`
	`MyFieldCount1`=6; çünkü 'Initials' yalnızca tekil değerlere sahip.

Örneklerde kullanılan veriler:

```
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() zaten yüklü durumdaki bir tablonun içine bakar ve **match_field_name** alanında **match_field_value** değerinin ilk oluşuna karşılık gelen **field_name** değerini döndürür. Bu tablo, mevcut tablo ya da daha önce yüklenmiş başka bir tablo olabilir.

Söz Dizimi:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Döndürülen değerin gerekli olduğu alanın adı. Giriş değeri bir dize (örneğin, tırnak içine alınmış bir değişmez değer) olarak verilmelidir.
match_ field_name	**match_field_value** öğesinin aranacağı alanın adı. Giriş değeri bir dize (örneğin, tırnak içine alınmış bir değişmez değer) olarak verilmelidir.
match_ field_value	**match_field_name** alanında aranacak değer.

Bağımsız Değişken	Açıklama
table_name	Değerin aranacağı tablonun adı. Giriş değeri bir dize (örneğin, tırnak içine alınmış bir değişmez değer) olarak verilmelidir. **table_name** atlandığı takdirde geçerli tablo kabul edilir.

Tırnak içinde olmayan bağımsız değişkenler geçerli tabloya referansta bulunur. Diğer tablolara referansta bulunmak için bağımsız değişkeni tek tırnak içine alın.

Sınırlamalar:

Tablo, sıralamanın tam olarak tanımlanmadığı birleştirmeler gibi karmaşık işlemlerin bir sonucu olmadıkça, aramanın yapıldığı sıra yükleme sırasıdır. Hem **field_name** hem de **match_field_name** öğesi, **table_name** öğesi ile belirtilen aynı tablodaki alanlar olmalıdır.

Herhangi bir eşleşme bulunamazsa NULL sonucu döndürülür.

Örnek

Komut dosyası

Aşağıdaki kod örneğini oluşturmak için veri yükleme düzenleyicisinde aşağıdaki verileri satır içi yükleme olarak yükleyin.

```
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;
```

Görselleştirme oluşturma

Bir Qlik Sense sayfasında bir tablo görselleştirmesi oluşturun. Tabloya **ProductID**, **InvoiceID**, **CustomerID**, **Units** ve **CategoryID** alanlarını ekleyin.

Sonuç

Sonuç tablosu

ProductID	InvoiceID	CustomerID	Birimler	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

Açıklama

Örnek verilerde **Lookup()** fonksiyonu şu biçimde kullanılmaktadır:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

ProductList tablosu ilk olarak yüklenir.

Lookup() fonksiyonu, **OrderData** tablosunu oluşturmak için kullanılır. Üçüncü bağımsız değişkeni **ProductID** olarak belirtir. Bu, tek tırnak içine alınmasıyla gösterildiği üzere **ProductList** içinde ikinci bağımsız değişkende (**'ProductID'**) değeri aranacak alandır.

Fonksiyon, **CategoryID** olarak yüklenen '**Category**' değerini döndürür (**ProductList** tablosunda).

drop deyimi, gerekli olmadığı için **ProductList** tablosunu veri modelinden siler; geriye **OrderData** sonuç tablosu kalır.

Lookup() fonksiyonu esnektir ve daha önce yüklenmiş herhangi bir tabloya erişim sağlayabilir. Bununla birlikte, Applymap() fonksiyonu ile karşılaştırıldığında yavaştır.

Ayrıca bkz.

ApplyMap (page 695)

NoOfRows - grafik fonksiyonu

NoOfRows(), bir tablodaki geçerli sütun segmentinde bulunan satırların sayısını döndürür. **NoOfRows()**, bit eşlem grafikleri için grafiğin düz tablo eşdeğerindeki satır sayısını döndürür.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Söz Dizimi:

```
NoOfRows([TOTAL])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Örnek: NoOfRows kullanan grafik ifadesi

Örnek - grafik ifadesi

Komut dosyası

Aşağıdaki grafik ifadesi örneklerini oluşturmak için aşağıdaki verileri veri yükleme düzenleyicisinde satır içi yükleme olarak yükleyin.

```
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 '|');
```

Grafik ifadesi

Qlik Sense sayfasında **Region** ve **SubRegion** alanlarını boyut olarak kullanarak bir tablo görselleştirmesi oluşturun. Hesaplama olarak `RowNo( ), NoOfRows()`, ve `NoOfRows(Total)` alanlarını ekleyin.

Sonuç

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

Açıklama

Bu örnekte, sırala düzeni ilk boyut olan Region'a göredir. Bunun sonucunda her sütun dilimi, örneğin, Afrika gibi aynı değere sahip bir grup bölgeden oluşur.

RowNo() sütunu, her sütun diliminin satır numaralarını gösterir; örneğin Afrika bölgesi için iki satır vardır. Daha sonra satır numaralandırması bir sonraki sütun segmenti (yani, Americas) için tekrar 1'den başlar.

NoOfRows() sütunu, her sütun dilimindeki satır sayısını sayar; örneğin sütun diliminde Avrupa'nın üç satırı vardır.

NoOfRows(Total) sütunu, `NoOfRows()` için `TOTAL` bağımsız değişkeni nedeniyle boyutları yoksayar ve tablodaki satırları sayar.

Tablo ikinci boyut olan SubRegion'a göre sıralandıysa, satır numaralandırmasının SubRegion'a göre değişmesi için, sütun dilimlerinde o boyut temel alınır.

Ayrıca bkz.

RowNo - grafik fonksiyonu (page 448)

Peek

Peek(), bir tablodaki bir alanın zaten yüklenmiş bir satırının değerini döndürür. Satır numarası belirtilebilir (tabloda olduğu gibi). Satır numarası belirtilmezse, daha önce yüklenmiş son kayıt kullanılır.

peek() fonksiyonu çoğu ez daha önce yüklenmiş bir tablonun ilgili sınırlarını; yani belirli bir alanın ilk ve son değerlerini bulmak için kullanılır. Çoğu durumda bu değer daha sonra örneğin bir do-while döngüsünde kullanılmak üzere bir değişkende saklanır.

Söz Dizimi:

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Döndürülen değerin gerekli olduğu alanın adı.Giriş değeri bir dize (örneğin, tırnak içine alınmış bir değişmez değer) olarak verilmelidir.
row_no	Tabloda alanın zorunlu olduğunu belirten satır. Bir ifade olabilir, ancak tamsayıya çözümlenmelidir. 0 ilk kaydı ve 1 ikinci kaydı gösterir ve bu böyle devam eder. Negatif sayılar tablonun sonundan itibaren sırayı belirtir. -1 değeri, okunan son kaydı gösterir. **row_no** belirtilmezse -1 olduğu varsayılır.
table_name	Sonunda iki nokta üst üste olmayan tablo etiketi. **table_name** belirtilmezse geçerli tablo olduğu varsayılır. **LOAD** deyimi dışında kullanılırsa veya başka bir tabloya referansta bulunursa, **table_name** dahil edilmelidir.

Sınırlamalar:

Fonksiyon yalnızca zaten yüklenmiş olan kayıtlardan değerler döndürebilir. Bu; bir tablonun ilk satırında row_no olarak -1 kullanan bir çağrı NULL döndürür.

Örnekler ve sonuçlar:

1. Örnek

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Sonuç tablosu

Çalışan kodu	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; çünkü `Peek('EmployeeCode',0, 'EmployeeDates')` öğesi EmployeeDates tablosundaki ilk EmployeeCode değerini döndürür.

LastCode = 106 çünkü `Peek('EmployeeCode',-1, 'EmployeeDates')`, EmployeeDates tablosundaki son EmployeeCode değerini döndürür.

row_no bağımsız değişkeninin değeri değiştirildiğinde tablodaki diğer satırların değerleri döndürülür. Şöyle ki:

`Peek('EmployeeCode',2, 'EmployeeDates')`, FirstCode olarak tablodaki 103 olan üçüncü değeri döndürür.

Ancak bu örneklerde tablonun üçüncü bağımsız değişken; **table_name** olarak belirtilmemesi durumunda fonksiyonun mevcut (bu örnekte dahili) tabloya başvurduğuna dikkat edin.

2. Örnek

Bir tabloda daha alttaki verilere erişmek istiyorsanız, bunu iki adımda yapmanız gerekir: Önce tüm tabloyu geçici bir tabloya yükleyin ve ardından **Peek()** kullanarak yeniden sıralayın.

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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.

Sonuç tablosu

Kimlik	Liste	Değer
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

IF() deyimi T1 geçici tablosundan oluşturulur.
`Peek('ID')`, geçerli tablo T2 içinde bir önceki satırda bulunan ID alanına referansta bulunur.
`Peek('List')`, ifade değerlendirildiği sırada oluşturulmakta olan T2 tablosunda bir önceki satırda bulunan List alanına referansta bulunur.

Deyim şöyle değerlendirilir:
Mevcut ID değeri, önceki ID değeriyle aynıysa, Peek('List') değerini mevcut Value değeri ile birleştirilmiş olarak yazın. Aksi takdirde sadece mevcut Value değerini yaz.

Peek('List') zaten birleştirilmiş bir sonuç içeriyorsa, yeni Peek('List') sonucu buna birleştirilir.

> ***Order by*** *cümlesine dikkat edin. Bu cümle tablonun nasıl sıralandığını belirtir (ID alanına göre artan sırada). Bu olmadan, Peek() fonksiyonu dahili tablonun rastgele düzenlemesini kullanır ve bu da öngörülemez sonuçlara yol açabilir.*

Örnek 3

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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];
```

Sonuç tablosu

Amount	AmountMonthBefore	Ay
1	4	2022-06
2	-	2022-01

Amount	AmountMonthBefore	Ay
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

AmountMonthBefore alanı, önceki ayın tutarını içerir.

Burada, row_no ve table_name parametreleri çıkarıldığından varsayılan değerler kullanılmaktadır. Bu örnekte, aşağıdaki üç fonksiyon çağrısı eşdeğerdir:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

row_no değeri olarak -1 kullanılması, önceki satırdaki değerin kullanılacağı anlamına gelir. Bu değer değiştirilerek tablodaki başka satırların değerleri getirilebilir:

Peek(Amount,2) tablodaki 7 olan üçüncü değeri döndürür.

Örnek 4

Doğru sonucun alınması için verilerin doğru sıralanması gerekir ancak maalesef durum her zaman böyle değildir. Dahası, Peek() fonksiyonu henüz yüklenmemiş verilere başvurmak için kullanılamaz. Geçici tablolar kullanarak ve verinin üzerinden birden çok kez geçerek bu tür sorunlardan kaçınılabilir.

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Açıklama

İlk tablo aya göre sıralandığından peek() fonksiyonu çoğu durumda yanlış ürünün tutarını döndürecektir. Bu nedenle tablonun yeniden sıralanması gerekir. Bu, verinin üzerinden ikinci bir kez daha geçip tmp2Amounts adlı yeni bir tablo oluşturularak yapılır. Order By cümlesine dikkat edin. Ürünleri önce ürüne göre, sonra ayar göre artan düzende sıralar.

If() fonksiyonu gereklidir çünkü AmountMonthBefore yalnızca önceki satır aynı ürünün önceki ay için verisini içeriyorsa hesaplanmalıdır. Geçerli satırdaki ürün önceki satırdaki ürünle karşılaştırılarak bu koşul doğrulanabilir.

İkinci tablo oluşturulduğunda, birinci tablo tmp1Amounts bir Drop Table cümlesi kullanılarak bırakılır.

Son olarak, verilerin üzerinden üçüncü bir kez daha, ancak bu kez aylar ters düzende sıralanmış olarak geçilir. Bu şekilde, AmountMonthAfter da hesaplanabilir.

Order By cümleleri tablonun nasıl sıralanacağını belirtir; bunlar olmadan Peek() fonksiyonu dahili tabloda hangi rasgele sıralama varsa onu kullanır ve bu beklenmedik sonuçlar ortaya çıkarabilir.

Sonuç

Sonuç tablosu

Ay	Product	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5
2022-01	A	5	6	7
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8
2022-05	A	4	7	-
2022-06	B	8	6	-

Örnek 5

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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;
```

Sonuç

Sonuç tablosu

Çeyrek	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

Açıklama

Yükleme ifadesi **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal**, önceki değerlerin geçerli değere eklendiği yinelemeli bir çağrı içerir. Bu işlem, komut dosyasındaki bir değer toplamını hesaplamak için kullanılır.

Ayrıca bkz.

Previous

Previous(), **where** cümlesi nedeniyle atılmamış önceki bir giriş kaydındaki verileri kullanarak **expr** ifadesinin değerini bulur. Bir iç tablonun ilk kaydında, bu fonksiyon NULL sonucunu döndürür.

Söz Dizimi:

```
Previous(expr)
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan. İfade, daha gerideki kayıtlara erişmek için, iç içe geçen **previous()** fonksiyonlarını içerebilir. Veriler doğrudan giriş kaynağından getirilir ve böylece Qlik Sense içine yüklenmemiş alanlara referansta bulunulması da mümkün olur (yani, ilişkili veritabanına depolanmamış olsalar bile).

Sınırlamalar:

Bir dahili tablonun ilk kaydında, fonksiyon NULL sonucunu döndürür.

Örnek:

Komut dosyanıza aşağıdakileri girin

```
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 '|');
```

Previous() fonksiyonunu **Load** deyimi içinde kullanarak, mevcut Sales değerini önceki değer ile karşılaştırabilir ve üçüncü bir alanda (Increase) kullanabiliriz.

Sonuç tablosu

Ay	Sales	Artış
1	12	-
2	13	1
3	15	2

Ay	Sales	Artış
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 - grafik fonksiyonu

Top(), bir tabloda sütun segmentinin ilk (en üstteki) satırında bulunan bir ifadeyi değerlendirir. Hesaplandığı satır **offset** değerine göre değişir (varsa) ve varsayılan ayar en üst satırdır. Tablolar dışındaki grafikler için **Top()** değerlendirmesi, grafiğin düz tablo eşdeğerinde geçerli sütunun ilk satırı üzerinde yapılır.

Söz Dizimi:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
offset	1'den büyük bir **offset** n belirtildiğinde, ifadenin değerlendirmesi üst satırın altında n satır aşağı taşınır. Negatif offset sayısı belirtilmesi, **Top** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Bottom** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir parametre **count** değeri belirtildiğinde, fonksiyon, geçerli sütun segmentinin her son **count** satırı için bir adet olmak üzere bir **count** değerleri aralığı döndürür. Bu biçimde, fonksiyon herhangi bir özel aralık fonksiyonuna yönelik bir bağımsız değişken olarak kullanılabilir. *Aralık fonksiyonları (page 702)*
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Sütun segmenti, geçerli sıralama düzeninde boyutlar için aynı değerlere sahip olan ardışık hücreler kümesi olarak tanımlanır. Kayıtlar arası grafik fonksiyonları, eşdeğer düz tablo grafiğinde en sağdaki boyut hariç tutularak sütun segmentinde hesaplanır. Grafikte yalnızca bir boyut varsa veya TOTAL niteleyicisi belirtilirse, ifade tüm tablo genelinde değerlendirilir.

Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sınırlamalar:

Yinelemeli çağrılar NULL sonucunu döndürür.

Örnekler ve sonuçlar:

Örnek: 1

Bu örnekte gösterilen tablonun ekran görüntüsünde, tablo görselleştirmesi **Customer** boyutundan ve şu hesaplamalardan oluşturulmuştur: `Sum(Sales)` ve `Top(Sum(Sales))`.

Top(Sum(Sales)) sütunu tüm satırlar için 587 döndürür; çünkü üst satırın değeri budur: **Astrida**.

Tabloda ayrıca, biri `Sum(Sales)+Top(Sum(Sales))` ifadesinden oluşturulan ve biri de **Top offset 3** etiketli (`Sum(Sales)+Top(Sum(Sales), 3)` ifadesi kullanılarak oluşturulmuş ve **offset** bağımsız değişkeni 3 olarak ayarlanmış) olmak üzere daha karmaşık hesaplamalar gösterilmektedir. Geçerli satıra ilişkin **Sum(Sales)** değerini üst satırdan itibaren üçüncü satırdan gelen değere ekler (yani, geçerli satır artı **Canutility** değeri).

1. Örnek

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

Örnek: 2

Bu örnekte gösterilen tabloların ekran görüntülerinde görselleştirmelere daha çok boyut eklenmiştir: **Month** ve **Product**. Birden fazla boyutu olan grafikler için **Above**, **Below**, **Top** ve **Bottom** fonksiyonlarını içeren ifadelerin sonuçları, sütun boyutlarının Qlik Sense tarafından sıralanma düzenine göre değişir. Qlik Sense, en son sıralanan boyuttan kaynaklanan sütun segmentlerini temel alarak fonksiyonları değerlendirir. Sütun sıralama düzeni, **Sıralama** altındaki özellikler panelinde kontrol edilir ve sütunların tabloda göründüğü düzen olmayabilir.

Örnek 2 için birinci tablo. Month (Jan) esas alındığında First value hesaplaması için Top değeri.

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

Örnek 2 için ikinci tablo. Product (Astrida için AA) esas alındığında First value hesaplaması için Top değeri.

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

Daha fazla ayrıntı için lütfen **Above** fonksiyonundaki 2. örneğe bakın.

Örnek 3

Örnek: 3	Sonuç
Top fonksiyonu, aralık fonksiyonları için giriş olarak kullanılabilir. Örneğin: `RangeAvg (Top(Sum (Sales),1,3))`.	**Top()** fonksiyonuna ait bağımsız değişkenlerde offset, 1 ve count olarak ayarlıdır. Fonksiyon, sütun segmentinde alt satırın altındaki satırdan başlayarak (çünkü offset=1) üç satırda ve bunun üstündeki iki satırda (satır varsa) **Sum(Sales)** ifadesinin sonuçlarını bulur. Bu üç değer, sağlanan sayı aralığındaki değerlerin ortalamasını bulan RangeAvg() fonksiyonu için giriş olarak kullanılır. Boyut olarak **Customer** öğesini içeren bir tablo RangeAvg() ifadesi için aşağıdaki sonuçları verir.
	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 '|');
```

Ayrıca bkz.

- *Bottom - grafik fonksiyonu (page 659)*
- *Above - grafik fonksiyonu (page 651)*
- *Sum - grafik fonksiyonu (page 231)*
- *RangeAvg (page 705)*
- *Aralık fonksiyonları (page 702)*

SecondaryDimensionality - grafik fonksiyonu

SecondaryDimensionality(), toplama olmayan içeriğe sahip (yani, kısmi toplamlar veya daraltılmış toplamalar içermeyen) boyut pivot tablo satırlarının sayısını döndürür. Bu fonksiyon, yatay pivot tablo boyutlarına yönelik **dimensionality()** fonksiyonuyla eşdeğerdir.

Söz Dizimi:

```
SecondaryDimensionality( )
```

Dönüş verileri türü: tamsayı

Sınırlamalar:

Pivot tablolarda kullanılmadığı sürece, **SecondaryDimensionality** fonksiyonu her zaman 0 döndürür.

After - grafik fonksiyonu

After(), pivot tablodaki bir satır segmenti içinde bulunan geçerli sütundan sonraki sütunda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür.

Söz Dizimi:

```
after([TOTAL] expr [, offset [, count ]])
```

Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.

Bağımsız Değişken	Açıklama
offset	1'den büyük bir **offset** n belirtildiğinde, ifadenin değerlendirmesi, geçerli satırdan sağa doğru n satır taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, **After** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Before** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** parametresi belirtildiğinde, fonksiyon ilk hücreden sağa doğru sayarak **count** değerine ulaşana kadar her tablo satırı için bir adet olmak üzere bir değer aralığı döndürür.
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Bir satır segmentinin son sütununda, bundan sonra gelen bir sütun olmadığından, bir NULL değeri döndürülür.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir. Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnek:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))`, geçerli sütunun hemen sağındaki üç sütunda değerlendirilen **sum(x)** fonksiyonunun üç sonucunun ortalamasını döndürür.

Before - grafik fonksiyonu

Before(), pivot tablodaki bir satır segmenti içinde bulunan geçerli sütundan önceki sütunda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür.

Söz Dizimi:

```
before([TOTAL] expr [, offset [, count]])
```

Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
offset	1'den büyük bir **offset** n belirtildiğinde, ifadenin değerlendirmesi, geçerli satırdan sola doğru n satır taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, **Before** fonksiyonunun karşılık gelen pozitif offset sayısı ile **After** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** parametresi belirtildiğinde, fonksiyon ilk hücreden sola doğru sayarak **count** değerine ulaşana kadar her tablo satırı için bir adet olmak üzere bir değer aralığı döndürür.
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Bir satır segmentinin ilk sütununda, bundan önce gelen bir sütun olmadığından, bir NULL değeri döndürülür.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnekler:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))`, geçerli sütunun hemen solundaki üç sütunda değerlendirilen **sum(x)** fonksiyonunun üç sonucunun ortalamasını döndürür.

First - grafik fonksiyonu

First(), pivot tablodaki geçerli satır segmentinin ilk sütununda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür. Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

Söz Dizimi:

```
first([TOTAL] expr [, offset [, count]])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expression	Hesaplanacak verileri içeren ifade veya alan.
offset	1'den büyük bir **offset** n belirtildiğinde, ifadenin değerlendirmesi, geçerli satırdan sağa doğru n satır taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, **First** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Last** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** parametresi belirtildiğinde, fonksiyon ilk hücreden sağa doğru sayarak **count** değerine ulaşana kadar her tablo satırı için bir adet olmak üzere bir değer aralığı döndürür.
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnekler:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` geçerli satır segmentinin en solundaki beş sütunda değerlendirilen **sum(x)** fonksiyonunun sonuçlarının ortalamasını döndürür.

Last - grafik fonksiyonu

Last(), pivot tablodaki geçerli satır segmentinin son sütununda göründüğü şekilde, pivot tablonun boyut değerleriyle değerlendirilen bir ifadenin değerini döndürür. Bu fonksiyon, pivot tablolar hariç tüm grafik türlerinde NULL değerini döndürür.

Söz Dizimi:

```
last([TOTAL] expr [, offset [, count]])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
offset	1'den büyük bir **offset** n belirtildiğinde, ifadenin değerlendirmesi, geçerli satırdan sola doğru n satır taşınır. Offset 0 olarak belirtildiğinde, ifade geçerli satır üzerinde değerlendirilir. Negatif offset sayısı belirtilmesi, **First** fonksiyonunun karşılık gelen pozitif offset sayısı ile **Last** fonksiyonu gibi çalışmasını sağlar.
count	1'den büyük üçüncü bir **count** parametresi belirtildiğinde, fonksiyon ilk hücreden sola doğru sayarak **count** değerine ulaşana kadar her tablo satırı için bir adet olmak üzere bir değer aralığı döndürür.
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnek:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` ifadesi, geçerli satır segmentinin en sağındaki beş sütunda değerlendirilen **sum(x)** fonksiyonunun sonuçlarının ortalamasını döndürür.

ColumnNo - grafik fonksiyonu

ColumnNo(), bir pivot tablodaki geçerli satır segmentinde bulunan geçerli sütunun sayısını döndürür. İlk sütunun sayısı 1'dir.

Söz Dizimi:

```
ColumnNo([total])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnek:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - grafik fonksiyonu

NoOfColumns(), bir pivot tablodaki geçerli satır segmentinde bulunan sütun sayısını döndürür.

Söz Dizimi:

```
NoOfColumns([total])
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
TOTAL	Tablo tek boyutluysa veya **TOTAL** niteleyicisi bir bağımsız değişken olarak kullanılıyorsa, geçerli sütun segmenti her zaman sütunun tamamına eşittir.

Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.Pivot tablolardaki yatay boyutlara yönelik alanlar arası sıralama düzeni, üstten alta doğru boyutların sırasıyla tanımlanır.

Örnek:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Mantıksal fonksiyonlar

Bu bölümde, mantıksal işlemleri ele alan fonksiyonlar açıklanmaktadır. Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

IsNum

İfade bir sayı olarak yorumlanabilirse, -1 (True), aksi takdirde 0 (False) değerini döndürür.

```
IsNum( expr )
```

IsText

İfade bir metin temsiline sahipse, -1 (True), aksi takdirde 0 (False) değerini döndürür.

```
IsText( expr )
```

*İfade NULL ise, hem **IsNum** hem de **IsText** 0 döndürür.*

Örnek:

Aşağıdaki örnek, metin değerlerinin ve sayısal değerlerin karma olarak bulunduğu bir satır içi tabloyu yükler ve değerin bir sayısal değer mi yoksa metin değeri mi olduğunu kontrol etmek üzere sırasıyla iki alan ekler.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Elde edilen tablo şöyle görünür:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Eşleme fonksiyonları

Bu bölümde, eşleme tablolarını kullanmaya yönelik fonksiyonlar açıklanmaktadır. Eşleme tabloları, kod yürütme sırasında alan değerlerini veya alan adlarını değiştirmek için kullanılabilir.

Eşleme fonksiyonları yalnızca veri kod dosyasında kullanılabilir.

Eşleme fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

ApplyMap
ApplyMap kod fonksiyonu, bir ifadenin çıkışını daha önceden yüklenmiş bir eşleme tablosuna eşlemek için kullanılır.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring
MapSubstring kod fonksiyonu herhangi bir ifadenin parçalarını daha önce yüklenmiş bir eşleme tablosuna eşlemek için kullanılır. Eşleme büyük/küçük harf duyarlıdır ve yinelemesizdir ve alt dizeler soldan sağa eşlenir.

```
MapSubstring ('mapname', expr)
```

ApplyMap

ApplyMap kod fonksiyonu, bir ifadenin çıkışını daha önceden yüklenmiş bir eşleme tablosuna eşlemek için kullanılır.

Söz Dizimi:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
map_name	Daha önce **mapping load** veya **mapping select** deyimi aracılığıyla oluşturulmuş bir eşleme tablosunun adı. Adı düz tek tırnak işaretleri içine alınmalıdır. *Bu fonksiyonu bir makro genişletilmiş değişkende kullanırsanız ve var olmayan bir eşleme tablosuna referans verirseniz fonksiyon çağrısı başarısız olur ve alan oluşturulmaz.*
expression	Sonucunun eşlenmesi gereken ifade.
default_ mapping	Belirtilirse bu değer, eşleme tablosunun expression için eşlenen bir değer içermemesi halinde varsayılan değer olarak kullanılır. Belirtilmezse, expression değeri olduğu gibi döndürülür.

ApplyMap çıktı alanı, girdi alanlarından biriyle aynı ada sahip olmamalıdır. Bu, beklenmeyen sonuçlara neden olabilir. Nasıl kullanılmaması gerektiğine dair örnek: `ApplyMap('Map', A) as A`.

Örnek:

Bu örnekte, ikamet ettikleri ülkeyi temsil eden ülke koduyla birlikte satış elemanlarının yer aldığı bir listeyi yüklüyoruz. Ülke kodunun yerine ülke adını koymak için, ülke kodunu ülkeyle eşleyen bir tablo kullanıyoruz. Eşleme tablosunda yalnızca üç ülke tanımlanmakta ve diğer ülke kodları 'Rest of the world' ile eşlenmektedir.

```
// 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';
```

Elde edilen tablo (Salespersons) şöyle görünür:

Resulting table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark

Salesperson	Country
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

MapSubstring

MapSubstring kod fonksiyonu herhangi bir ifadenin parçalarını daha önce yüklenmiş bir eşleme tablosuna eşlemek için kullanılır. Eşleme büyük/küçük harf duyarlıdır ve yinelemesizdir ve alt dizeler soldan sağa eşlenir.

Söz Dizimi:

```
MapSubstring('map_name', expression)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
map_name	Bir **mapping load** veya **mapping select** deyimi ile daha önce okunmuş bir eşleme tablosunun adı. Ad, düz tek tırnak işaretleri içine alınmalıdır.  *Bu fonksiyonu bir makro genişletilmiş değişkende kullanırsanız ve var olmayan bir eşleme tablosuna referans verirseniz fonksiyon çağrısı başarısız olur ve alan oluşturulmaz.*
expression	Sonucu alt dizeler ile eşlenecek ifade.

Örnek:

Bu örnekte ürün modellerinin listesini yüklüyoruz. Her modelin bileşik bir kod ile açıklanan bir öznitelik kümesi vardır. MapSubstring ile eşleme tablosunu kullanarak öznitelik kodlarını bir açıklamaya genişletebiliriz.

```
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';
```

Elde edilen tablo şöyle görünür:

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 Matematiksel fonksiyonlar

Bu bölümde, matematiksel sabitlere ve Boole değerlerine yönelik fonksiyonlar açıklanmaktadır. Bu fonksiyonların parametresi yoktur; ancak parantezler yine de gereklidir.

Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

e

Fonksiyon, doğal logaritmaların tabanını döndürür: **e** (2,71828...).

```
e( )
```

false

Fonksiyon, ifadelerde mantıksal yanlış olarak kullanılabilen ve metin değeri 'False' iken sayısal değeri 0 olan bir ikili değer döndürür.

```
false( )
```

pi

Fonksiyon π değerini (3,14159...) döndürür.

```
pi( )
```

rand

Fonksiyon, 0 ile 1 arasında rastgele bir sayı döndürür. Bu, örnek veriler oluşturmak için kullanılabilir.

```
rand( )
```

Örnek:

Bu örnek kod, rastgele seçilmiş büyük harf karakterlerini, yani 65 ila 91 aralığındaki (65+26) karakterleri içeren 1000 kayıtlık bir tablo oluşturur.

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true

Fonksiyon, ifadelerde mantıksal yanlış olarak kullanılabilen ve metin değeri 'True' iken sayısal değeri -1 olan bir ikili değer döndürür.

```
true( )
```

5.20 NULL fonksiyonları

Bu bölümde, NULL değerler döndürmeye veya bu değerleri algılamaya yönelik fonksiyonlar açıklanmaktadır.

Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

NULL fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

EmptyIsNull

EmptyIsNull fonksiyonu boş dizeleri NULL öğesine dönüştürür. Bu nedenle, parametre boş bir dize ise NULL döndürür, boş değilse parametreyi döndürür.

```
EmptyIsNull (expr )
```

IsNull

IsNull fonksiyonu, bir ifadenin değerinin NULL olup olmadığını test eder; öyleyse -1 (True) döndürür, aksi takdirde 0 (False) döndürür.

```
IsNull (expr )
```

Null

Null fonksiyonu bir NULL değer döndürür.

```
NULL( )
```

EmptyIsNull

EmptyIsNull fonksiyonu boş dizeleri NULL öğesine dönüştürür. Bu nedenle, parametre boş bir dize ise NULL döndürür, boş değilse parametreyi döndürür.

Söz Dizimi:

```
EmptyIsNull(exp )
```

Örnekler ve sonuçlar:

Kod örnekleri

Örnek	Sonuç
`EmptyIsNull(AdditionalComments)`	Bu ifade, boş dizeler yerine *AdditionalComments* alanının boş dize değerlerini null olarak döndürür. Boş olmayan dizeler ve sayılar döndürülür.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Bu ifade, *PhoneNumber* alanındaki tüm çizgi, boşluk ve parantezleri kaldıracaktır. Hiç karakter kalmadıysa, EmptyIsNull fonksiyonu boş dizeyi null olarak döndürür; boş bir telefon numarası, telefon numarası olmamasıyla aynı şeydir.

IsNull

IsNull fonksiyonu, bir ifadenin değerinin NULL olup olmadığını test eder; öyleyse -1 (True) döndürür, aksi takdirde 0 (False) döndürür.

Söz Dizimi:

```
IsNull(expr )
```

*Sıfır uzunluklu bir dize NULL olarak değerlendirilmez ve **IsNull** deyiminin False sonucunu döndürmesine neden olur.*

Örnek: Veri kod dosyası

Bu örnekte, ilk üç satırı - sütununda hiçbir şey içermeyen ya da 'NULL' veya Value değerlerini içeren dört satırlı bir satır içi tablo yüklenmektedir. **Null** fonksiyonunu kullanarak orta öncelikli **LOAD** ile bu değerleri doğru NULL değer temsillerine dönüştürüyoruz.

İlk öncelikli **LOAD** deyimi, **IsNull** fonksiyonunu kullanmak suretiyle değerin NULL olup olmadığını kontrol ederek bir alan ekler.

```
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];
```

Sonuçta ortaya çıkan tablo budur. ValueNullConv sütununda NULL değerler - ile temsil edilmektedir.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Null fonksiyonu bir NULL değer döndürür.

Söz Dizimi:

```
Null( )
```

Örnek: Veri kod dosyası

Bu örnekte, ilk üç satırı - sütununda hiçbir şey içermeyen ya da 'NULL' veya Value değerlerini içeren dört satırlı bir satır içi tablo yüklenmektedir. Bu değerleri doğru NULL değer temsillerine dönüştürmek istiyoruz.

Ortadaki öncelikli **LOAD** bu dönüşümü **Null** fonksiyonunu kullanarak yapar.

İlk öncelikli **LOAD** bir alan ekleyerek değerin NULL olup olmadığını kontrol eder (bu örnekte yalnızca gösterim amaçlıdır).

```
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];
```

Sonuçta ortaya çıkan tablo budur. ValueNullConv sütununda NULL değerler - ile temsil edilmektedir.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Aralık fonksiyonları

Aralık fonksiyonları, bir değer dizisi alan ve sonuç olarak tek bir değer üreten fonksiyonlardır. Tüm aralık fonksiyonları hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

Örneğin bir görselleştirmede, aralık fonksiyonu bir kayıtlar arası dizisinden tek bir değer hesaplayabilir. Veri kod dosyasında aralık fonksiyonu, iç tablodaki bir değer dizisinden tek bir değer hesaplayabilir.

Aralık fonksiyonları, artık eski olarak değerlendirilmesi gereken şu genel sayısal fonksiyonların yerini alır: ***numsum****,****numavg****,* ***numcount****,****nummin*** *ve* ***nummax****.*

Temel aralık fonksiyonları

RangeMax

RangeMax(), ifadede veya alanda bulunan en yüksek sayısal değerleri döndürür.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString(), ifadede veya alanda bulduğu metin sıralama düzenindeki son değeri döndürür.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin(), ifade veya alan dahilinde bulunan en düşük sayısal değerleri döndürür.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString(), ifadede veya alanda bulduğu metin sıralama düzenindeki ilk değeri döndürür.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode(), ifadede veya alanda en yaygın olarak geçen değeri (mod değeri) bulur.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly(), ifade tek bir benzersiz değer olarak değerlendirilirse bir değer döndüren ikili fonksiyon olur. Böyle bir durum söz konusu değilse **NULL** döndürür.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum(), değer aralıkları toplamını döndürür. Tüm sayısal olmayan değerler 0 olarak işlenir.

```
RangeSum (first_expr[, Expression])
```

Sayaç aralık fonksiyonları

RangeCount

RangeCount() ifadedeki veya alandaki değerlerin (hem metin hem de sayısal) sayısını döndürür.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount(), ifadede veya alanda sayısal olmayan değerlerin (NULL dahil) sayısını döndürür.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount(), ifadede veya alanda NULL değerlerin sayısını bulur.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount(), bir ifadede veya alanda sayısal değerlerin sayısını bulur.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount(), bir ifadede veya alanda metin değerlerinin sayısını döndürür.

```
RangeTextCount (first_expr[, Expression])
```

İstatistiksel aralık fonksiyonları

RangeAvg

RangeAvg() aralık ortalamasını döndürür. Fonksiyonun girdisi bir değer aralığı veya bir ifade olabilir.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel(), iki veri kümesi için korelasyon katsayısını döndürür. Korelasyon katsayısı veri kümeleri arasındaki ilişkinin bir hesaplamasıdır.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile(), bir sayı aralığının n. **fractile** değerine (yüzdelik dilim) karşılık gelen değeri döndürür.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis(), bir sayı aralığının basıklığına karşılık gelen değeri döndürür.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew(), bir sayı aralığının eğriliğine karşılık gelen değeri döndürür.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev(), bir sayı aralığının standart sapmasını bulur.

```
RangeStdev (expr1[, Expression])
```

Finansal aralık fonksiyonları

RangeIRR

RangeIRR(), giriş değerleri tarafından temsil edilen bir nakit akışları serisi için iç geri dönüş oranını döndürür.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV(), iskonto oranına ve gelecekteki düzenli ödemelerin (negatif değerlerin) ve gelirlerin (pozitif değerlerin) serisine dayalı olarak bir yatırımın net mevcut değerini döndürür. Sonuç **money** öğesinin varsayılan sayı biçimine sahiptir.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR(), dönemsel olması gerekmeyen nakit akışlarının planı için iç geri dönüş oranını döndürür. Bir dizi dönemsel nakit akışı için iç geri dönüş oranını hesaplamak için **RangeIRR** fonksiyonunu kullanın.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV(), dönemsel olması gerekmeyen nakit akışlarının planı için net mevcut değerini döndürür. Sonuç, para için varsayılan sayı biçimine sahiptir. Bir dizi dönemsel nakit akışı için net mevcut değeri hesaplamak için, **RangeNPV** fonksiyonunu kullanın.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Ayrıca bkz.

Kayıtlar arası fonksiyonlar (page 647)

RangeAvg

RangeAvg() aralık ortalamasını döndürür. Fonksiyonun girdisi bir değer aralığı veya bir ifade olabilir.

Söz Dizimi:

```
RangeAvg(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

Kod örnekleri

Örnekler	Sonuçlar
`RangeAvg (1,2,4)`	2,33333333 döndürür
`RangeAvg (1,'xyz')`	1 döndürür
`RangeAvg (null( ), 'abc')`	NULL döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeAvg değerlerini gösterir.

Sonuç tablosu

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

İfadeyi içeren örnek:

```
RangeAvg (Above(MyField),0,3))
```

Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan üç **MyField** değerinin aralık sonucunun hareketli ortalamasını döndürür. Üçüncü bağımsız değişkenin 3 olarak belirtilmesiyle, **Above()** fonksiyonu üstte yeterli satırın bulunduğu durumlarda üç değer döndürür ve bunlar da **RangeAvg()** fonksiyonu için giriş değeri olarak alınır.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	En üst satır bu olduğundan, aralık yalnızca tek bir değerden oluşur.
2	6	Bu satırın üzerinde sadece bir satır bulunduğundan aralık şöyle olur: 10,2.
8	6.6666666667	RangeAvg(10,2,8) eşdeğeri
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Ayrıca bkz.

- *Avg - grafik fonksiyonu (page 270)*
- *Count - grafik fonksiyonu (page 235)*

RangeCorrel

RangeCorrel(), iki veri kümesi için korelasyon katsayısını döndürür. Korelasyon katsayısı veri kümeleri arasındaki ilişkinin bir hesaplamasıdır.

Söz Dizimi:

```
RangeCorrel(x_value , y_value[, Expression])
```

Dönüş verileri türü: sayısal

Veri serisi (x,y) çift olarak girilmelidir. Örneğin, dizi 1 ve dizi 2 olmak üzere (dizi 1 = 2,6,9 ve dizi 2 = 3,8,4) iki veri serisini değerlendirmek için `RangeCorrel (2,3,6,8,9,4)` yazarsınız ve bu da 0,269 değerini döndürür.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
x-value, y-value	Her bir değer, üçüncü bir isteğe bağlı parametresi bulunan kayıtlar arası fonksiyonların döndürdüğü tek bir değeri ya da bir değer aralığını temsil eder. Her değer veya değer aralığı, bir **x-value** veya bir **y-values** aralığına karşılık gelmelidir.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Bu fonksiyonun hesaplanacak en az iki çift koordinata ihtiyacı vardır.

Metin değerleri, NULL değerleri ve eksik değerler NULL döndürür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeCorrel (2,3,6,8,9,4,8,5)`	0,2492 döndürür. Bu fonksiyon, kodda yüklenebilir ve ifade düzenleyicisinde görselleştirmeye eklenebilir.

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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 '|');
```

Boyut olarak ID1 ve hesaplama içeren bir tabloda:**RangeCorrel()** fonksiyonu olan RangeCorrel (x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)), ID1 değerlerinin her biri için altı adet x,y çiftli aralık üzerindeki **Correl** değerini bulur.

Sonuç tablosu

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Örnek:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

Boyut olarak RangeID ve hesaplama içeren bir tabloda:**RangeCorrel()** fonksiyonu olan RangeCorrel (Below(X,0,4,BelowY,0,4)), 4 olarak ayarlanan (count) üçüncü bağımsız değişken, yüklenen XY tablosundan dört adet x-y değerli bir aralık oluşturduğundan **Below()** fonksiyonlarının sonuçlarını kullanır.

Sonuç tablosu

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

RangeID 01 değeri, manuel olarak girilen RangeCorrel(2,3,6,8,9,4,8,5) ile aynıdır. Diğer RangeID değerleri için, Below() fonksiyonu tarafından üretilen dizi şöyledir: (6,8,9,4,8,5), (9,4,8,5) ve (8,5). Bu dizinin son öğesi null sonuç oluşturur.

Ayrıca bkz.

Correl - grafik fonksiyonu (page 272)

RangeCount

RangeCount() ifadedeki veya alandaki değerlerin (hem metin hem de sayısal) sayısını döndürür.

Söz Dizimi:

```
RangeCount(first_expr[, Expression])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Ölçülecek verileri içeren ifade veya alan.
Expression	Ölçülecek veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

NULL değerler sayılmaz.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeCount (1,2,4)`	3 döndürür
`RangeCount (2,'xyz')`	2 döndürür
`RangeCount (null( ))`	0 döndürür
`RangeCount (2,'xyz', null())`	2 döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeCount değerlerini gösterir.

Sonuçlar tablosu

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

İfadeyi içeren örnek:

```
RangeCount (Above(MyField,1,3))
```

Üç **MyField** sonucunda yer alan değerlerin sayısını döndürür. **Above()** fonksiyonunun birinci bağımsız değişkeni olarak 1 ve ikinci bağımsız değişkeni olarak 3 belirtildiğinde, geçerli satırın üzerindeki ilk üç alandan değerleri döndürür (yeterli satır bulunduğu durumda) ve bunlar da **RangeCount()** fonksiyonu için giriş değeri olarak alınır.

Örneklerde kullanılan veriler:

Örnek veriler

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Ayrıca bkz.

Count - grafik fonksiyonu (page 235)

RangeFractile

RangeFractile(), bir sayı aralığının n. **fractile** değerine (yüzdelik dilim) karşılık gelen değeri döndürür.

RangeFractile(), fraktili hesaplarken en yakın sıralamalar arasında doğrusal enterpolasyon kullanır.

Söz Dizimi:

```
RangeFractile(fractile, first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
fractile	Hesaplanacak fraktile (kesir olarak ifade edilen yüzdelik dilim) karşılık gelen, 0 ile 1 arasında bir sayı.
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeFractile (0.24,1,2,4,6)`	1,72 döndürür
`RangeFractile(0.5,1,2,3,4,6)`	3 döndürür
`RangeFractile (0.5,1,2,5,6)`	3,5 döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeFrac değerlerini gösterir.

Sonuç tablosu

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

İfadeyi içeren örnek:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

Bu örnekte, **Above()** kayıtlar arası fonksiyonu, isteğe bağlı offset ve count bağımsız değişkenlerini içermektedir. Böylece, aralık fonksiyonlarının herhangi biri için giriş olarak kullanılabilecek bir sonuç aralığı üretilir. Bu durumda, `Above(Sum(MyField),0,3)` fonksiyonu, geçerli satır ve üzerindeki iki satır için `MyField` değerlerini döndürür. Bu değerler **RangeFractile()** fonksiyonu için giriş değerlerini sağlar. Bu nedenle, aşağıdaki tablodaki alt satır için bu, `RangeFractile(0.5, 3,4,6)` eşdeğeridir; başka bir deyişle, 3, 4 ve 6 dizisi için 0,5 fraktilidir. Geçerli satırın yukarısında bir satır olmayacak şekilde, aşağıdaki tabloda yer alan ilk iki satır, aralıktaki değer sayısı uygun şekilde azaltılır. Diğer kayıtlar arası fonksiyonları için benzer sonuçlar üretilir.

Örnek veriler

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
```

```
3
4
5
6
] ;
```

Ayrıca bkz.

- *Above - grafik fonksiyonu (page 651)*
- *Fractile - grafik fonksiyonu (page 276)*

RangeIRR

RangeIRR(), giriş değerleri tarafından temsil edilen bir nakit akışları serisi için iç geri dönüş oranını döndürür.

Dahili geri dönüş oranı, düzenli aralıklarda meydana gelen ödemelerden (negatif değerlerden) ve gelirden (pozitif değerlerden) oluşan ve bir yatırım için alınan faiz oranıdır.

Söz Dizimi:

```
RangeIRR(value[, value][, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Üçüncü bir isteğe bağlı parametresi bulunan kayıtlar arası fonksiyonun döndürdüğü tek bir değer veya bir değer aralığı. Bu fonksiyonun hesaplanacak en az bir pozitif ve bir negatif değeri olması gerekir.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnek tablo

Örnekler	Sonuçlar
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	0,0866 döndürür

Örnekler	Sonuçlar

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.
`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 '|');`
	Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen RangeIRR değerlerini gösterir.
	RangeID — RangeIRR
	1 — 0.0639
	2 — 0.8708
	3 — -
	4 — 5.8419
	5 — 0.9318
	6 — -0.2566

Ayrıca bkz.

Kayıtlar arası fonksiyonlar (page 647)

RangeKurtosis

RangeKurtosis(), bir sayı aralığının basıklığına karşılık gelen değeri döndürür.

Söz Dizimi:

```
RangeKurtosis(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeKurtosis (1,2,4,7)`	-0,28571428571429 döndürür

Ayrıca bkz.

Kurtosis - grafik fonksiyonu (page 284)

RangeMax

RangeMax(), ifadede veya alanda bulunan en yüksek sayısal değerleri döndürür.

Söz Dizimi:

```
RangeMax(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMax (1,2,4)`	4 döndürür
`RangeMax (1,'xyz')`	1 döndürür
`RangeMax (null( ), 'abc')`	NULL döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeMax değerlerini gösterir.

Sonuç tablosu

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

İfadeyi içeren örnek:

```
RangeMax (Above(MyField,0,3))
```

Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan üç **MyField** değeri aralığında maksimum değeri döndürür. Üçüncü bağımsız değişkenin 3 olarak belirtilmesiyle, **Above()** fonksiyonu üstte yeterli satırın bulunduğu durumlarda üç değer döndürür ve bunlar da **RangeMax()** fonksiyonu için giriş değeri olarak alınır.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString(), ifadede veya alanda bulduğu metin sıralama düzenindeki son değeri döndürür.

Söz Dizimi:

```
RangeMaxString(first_expr[, Expression])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMaxString (1,2,4)`	4 döndürür
`RangeMaxString ('xyz','abc')`	'xyz' döndürür
`RangeMaxString (5,'abc')`	'abc' döndürür
`RangeMaxString (null( ))`	NULL döndürür

İfadeyi içeren örnek:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MaxString(MyField)** fonksiyonunun üç sonucundan sonuncusunu (metin sıralama düzeninde) döndürür.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Ayrıca bkz.

MaxString - grafik fonksiyonu (page 402)

RangeMin

RangeMin(), ifade veya alan dahilinde bulunan en düşük sayısal değerleri döndürür.

Söz Dizimi:

```
RangeMin(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMin (1,2,4)`	1 döndürür
`RangeMin (1,'xyz')`	1 döndürür
`RangeMin (null( ), 'abc')`	NULL döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeMin değerlerini gösterir.

Sonuç tablosu

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

İfadeyi içeren örnek:

```
RangeMin (Above(MyField,0,3)
```

Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan üç **MyField** değeri aralığında minimum değeri döndürür. Üçüncü bağımsız değişkenin 3 olarak belirtilmesiyle, **Above()** fonksiyonu üstte yeterli satırın bulunduğu durumlarda üç değer döndürür ve bunlar da **RangeMin()** fonksiyonu için giriş değeri olarak alınır.

Örneklerde kullanılan veriler:

Örnek veriler

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Ayrıca bkz.

Min - grafik fonksiyonu (page 222)

RangeMinString

RangeMinString(), ifadede veya alanda bulduğu metin sıralama düzenindeki ilk değeri döndürür.

Söz Dizimi:

```
RangeMinString(first_expr[, Expression])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMinString (1,2,4)`	1 döndürür
`RangeMinString ('xyz','abc')`	'abc' döndürür
`RangeMinString (5,'abc')`	5 döndürür
`RangeMinString (null( ))`	NULL döndürür

İfadeyi içeren örnek:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MinString(MyField)** fonksiyonunun üç sonucundan ilkini (metin sıralama düzeninde) döndürür.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
```

```
'xyz'
9
] ;
```

Ayrıca bkz.

MinString - grafik fonksiyonu (page 405)

RangeMissingCount

RangeMissingCount(), ifadede veya alanda sayısal olmayan değerlerin (NULL dahil) sayısını döndürür.

Söz Dizimi:

```
RangeMissingCount(first_expr[, Expression])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Ölçülecek verileri içeren ifade veya alan.
Expression	Ölçülecek veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMissingCount (1,2,4)`	0 döndürür
`RangeMissingCount (5,'abc')`	1 döndürür
`RangeMissingCount (null( ))`	1 döndürür

İfadeyi içeren örnek:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MinString(MyField)** fonksiyonunun üç sonucunda sayısal olmayan değerlerin sayısını döndürür.

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Bu satırın üzerinde satır olmadığından 2 döndürür; 3 değerin 2'si eksiktir.
abc	2	Geçerli satır üzerinde yalnızca 1 satır olduğundan ve geçerli satır sayısal olmadığından ('abc') 2 döndürür.
8	1	3 satırdan 1'i sayısal olmayan bir değer ('abc') içerdiğinden 1 döndürür.
def	2	3 satırdan 2'si sayısal olmayan değerler ('def' ve 'abc') içerdiğinden 2 döndürür.
xyz	2	3 satırdan 2'si sayısal olmayan değerler (' xyz' ve 'def') içerdiğinden 2 döndürür.
9	2	3 satırdan 2'si sayısal olmayan değerler (' xyz' ve 'def') içerdiğinden 2 döndürür.

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Ayrıca bkz.

MissingCount - grafik fonksiyonu (page 239)

RangeMode

RangeMode(), ifadede veya alanda en yaygın olarak geçen değeri (mod değeri) bulur.

Söz Dizimi:

```
RangeMode(first_expr {, Expression})
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Birden fazla değer en yüksek sıklığı paylaşıyorsa, NULL döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeMode (1,2,9,2,4)`	2 döndürür
`RangeMode ('a',4,'a',4)`	NULL döndürür
`RangeMode (null( ))`	NULL döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen **MyRangeMode** değerlerini gösterir.

Sonuçlar tablosu

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

İfadeyi içeren örnek:

```
RangeMode (Above(MyField,0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MyField** fonksiyonunun üç sonucunda en yaygın olarak görülen değeri döndürür. Üçüncü bağımsız değişkenin 3 olarak belirtilmesiyle, **Above()** fonksiyonu üstte yeterli satırın bulunduğu durumlarda üç değer döndürür ve bunlar da **RangeMode()** fonksiyonu için giriş değeri olarak alınır.

Örnekte kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeMode(Above(MyField,0,3))
10	Üstte satır olmadığından 10 döndürür; tek değer en yaygın olarak görülen değerdir.
2	-
8	-
18	-
5	-
9	-

Ayrıca bkz.

Mode - grafik fonksiyonu (page 225)

RangeNPV

RangeNPV(), iskonto oranına ve gelecekteki düzenli ödemelerin (negatif değerlerin) ve gelirlerin (pozitif değerlerin) serisine dayalı olarak bir yatırımın net mevcut değerini döndürür. Sonuç **money** öğesinin varsayılan sayı biçimine sahiptir.

Mutlaka dönemsel olması gerekmeyen nakit akışları için bkz. *RangeXNPV (page 738)*.

Söz Dizimi:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	Dönem başına faiz oranı.
value	Her dönemin sonunda meydana gelen ödeme veya gelir. Her bir değer, üçüncü bir isteğe bağlı parametresi bulunan kayıtlar arası fonksiyonun döndürdüğü tek bir değer ya da bir değer aralığı olabilir.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Örnekler	Sonuçlar
`RangeNPV(0.1,-10000,3000,4200,6800)`	1188,44 döndürür

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.
`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 '|');`
	Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen RangeNPV değerlerini gösterir.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

Ayrıca bkz.

- *Kayıtlar arası fonksiyonlar (page 647)*

RangeNullCount

RangeNullCount(), ifadede veya alanda NULL değerlerin sayısını bulur.

Söz Dizimi:

```
RangeNullCount(first_expr [, Expression])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeNullCount (1,2,4)`	0 döndürür
`RangeNullCount (5,'abc')`	0 döndürür
`RangeNullCount (null( ), null( ))`	2 döndürür

İfadeyi içeren örnek:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **Sum(MyField)** fonksiyonunun üç sonucunda NULL değerlerin sayısını döndürür.

*Aşağıdaki örnekte **MyField** öğesinin kopyalanması NULL değeri ile sonuçlanmaz.*

Örnek veriler

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Bu satırın üzerinde satır olmadığından 2 döndürür; 3 değerin 2'si eksiktir (=NULL).
'abc'	Geçerli satırın üzerinde sadece bir satır olduğundan 1 döndürür; üç değerden birisi eksiktir (=NULL).
8	Üç satırdan hiçbiri NULL değeri olmadığından 0 döndürür.

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Ayrıca bkz.

NullCount - grafik fonksiyonu (page 241)

RangeNumericCount

RangeNumericCount(), bir ifadede veya alanda sayısal değerlerin sayısını bulur.

Söz Dizimi:

```
RangeNumericCount(first_expr[, Expression])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeNumericCount (1,2,4)`	3 döndürür
`RangeNumericCount (5,'abc')`	1 döndürür
`RangeNumericCount (null( ))`	0 döndürür

İfadeyi içeren örnek:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MaxString(MyField)** fonksiyonunun üç sonucunda sayısal değerlerin sayısını döndürür.

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
abc	1
8	2
def	1
xyz	1
9	1

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Ayrıca bkz.

NumericCount - grafik fonksiyonu (page 244)

RangeOnly

RangeOnly(), ifade tek bir benzersiz değer olarak değerlendirilirse bir değer döndüren ikili fonksiyon olur. Böyle bir durum söz konusu değilse **NULL** döndürür.

Söz Dizimi:

```
RangeOnly(first_expr[, Expression])
```

Dönüş verileri türü: ikili

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

Örnekler	Sonuçlar
`RangeOnly (1,2,4)`	NULL döndürür
`RangeOnly (5,'abc')`	NULL döndürür
`RangeOnly (null( ), 'abc')`	'abc' döndürür
`RangeOnly(10,10,10)`	10 döndürür

Ayrıca bkz.

Only - grafik fonksiyonu (page 228)

RangeSkew

RangeSkew(), bir sayı aralığının eğriliğine karşılık gelen değeri döndürür.

Söz Dizimi:

```
RangeSkew(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`rangeskew (1,2,4)`	0,93521952958283 döndürür
`rangeskew (above (SalesValue,0,3))`	Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan above() fonksiyonundan döndürülen üç değerin aralığının hareketli eğriliğini döndürür.

Örnekte kullanılan veriler:

Örnek veriler

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
] ;
```

Ayrıca bkz.

- *Skew - grafik fonksiyonu (page 316)*

RangeStdev

RangeStdev(), bir sayı aralığının standart sapmasını bulur.

Söz Dizimi:

```
RangeStdev(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

Sayısal değer bulunmuyorsa NULL sonucu döndürülür.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeStdev (1,2,4)`	1,5275252316519 döndürür
`RangeStdev (null( ))`	NULL döndürür
`RangeStdev (above (SalesValue),0,3))`	Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan above() fonksiyonundan döndürülen üç değerin aralığının hareketli standardını döndürür.

Örnekte kullanılan veriler:

Örnek veriler

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
] ;
```

Ayrıca bkz.

Stdev - grafik fonksiyonu (page 319)

RangeSum

RangeSum(), değer aralıkları toplamını döndürür. Tüm sayısal olmayan değerler 0 olarak işlenir.

Söz Dizimi:

```
RangeSum(first_expr[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Sınırlamalar:

RangeSum fonksiyonu sayısal olmayan tüm değerlerle 0 gibi işlem yapar.

Örnekler ve sonuçlar:

Örnekler

Örnekler	Sonuçlar
`RangeSum (1,2,4)`	7 döndürür
`RangeSum (5,'abc')`	5 döndürür
`RangeSum (null( ))`	0 döndürür

Örnek:

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.

```
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
];
```

Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen MyRangeSum değerlerini gösterir.

Sonuç tablosu

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

İfadeyi içeren örnek:

```
RangeSum (Above(MyField,0,3))
```

Geçerli satırda ve geçerli satırın iki satır üzerinde hesaplanan üç **MyField)** değerinin toplamını döndürür. Üçüncü bağımsız değişkenin 3 olarak belirtilmesiyle, **Above()** fonksiyonu üstte yeterli satırın bulunduğu durumlarda üç değer döndürür ve bunlar da **RangeSum()** fonksiyonu için giriş değeri olarak alınır.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Ayrıca bkz.

- *Sum - grafik fonksiyonu (page 231)*
- *Above - grafik fonksiyonu (page 651)*

RangeTextCount

RangeTextCount(), bir ifadede veya alanda metin değerlerinin sayısını döndürür.

Söz Dizimi:

```
RangeTextCount(first_expr[, Expression])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bu fonksiyonun bağımsız değişkenleri, bir değer listesi döndüren kayıtlar arası fonksiyonlar içerebilir.

Bağımsız Değişken

Bağımsız Değişken	Açıklama
first_expr	Hesaplanacak verileri içeren ifade veya alan.
Expression	Hesaplanacak veri aralığını içeren isteğe bağlı ifadeler veya alanlar.

Örnekler ve sonuçlar:

İşlev örnekleri

Örnekler	Sonuçlar
`RangeTextCount (1,2,4)`	0 döndürür
`RangeTextCount (5,'abc')`	1 döndürür
`RangeTextCount (null( ))`	0 döndürür

İfadeyi içeren örnek:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Geçerli satırda ve geçerli satırın üzerindeki iki satırda değerlendirilen **MaxString(MyField)** fonksiyonunun üç sonucunda metin değerlerinin sayısını döndürür.

Örneklerde kullanılan veriler:

*Örneğin beklendiği gibi çalışmasını sağlamak için **MyField** sıralamasını devre dışı bırakın.*

Örnek veriler

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

Örneklerde kullanılan veriler:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Ayrıca bkz.

TextCount - grafik fonksiyonu (page 247)

RangeXIRR

RangeXIRR(), dönemsel olması gerekmeyen nakit akışlarının planı için iç geri dönüş oranını döndürür. Bir dizi dönemsel nakit akışı için iç geri dönüş oranını hesaplamak için **RangeIRR** fonksiyonunu kullanın.

Söz Dizimi:

```
RangeXIRR(value, date{, value, date})
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Tarihler halinde ödeme planına karşılık gelen bir nakit akışı veya bir dizi nakit akışı. Değerler serisi en az bir pozitif ve bir negatif değer içermelidir.
date	Nakit akışı ödemelerine karşılık gelen bir ödeme tarihi veya ödeme tarihleri planı.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Örnekler	Sonuçlar
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	0,1532 döndürür

Ayrıca bkz.

RangeIRR (page 714)

RangeXNPV

RangeXNPV(), dönemsel olması gerekmeyen nakit akışlarının planı için net mevcut değerini döndürür. Sonuç, para için varsayılan sayı biçimine sahiptir. Bir dizi dönemsel nakit akışı için net mevcut değeri hesaplamak için, **RangeNPV** fonksiyonunu kullanın.

Söz Dizimi:

```
RangeXNPV(discount_rate, values, dates[, Expression])
```

Dönüş verileri türü: sayısal

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
discount_rate	Dönem başına faiz oranı.

Bağımsız Değişken	Açıklama
values	Tarihler halinde ödeme planına karşılık gelen bir nakit akışı veya bir dizi nakit akışı. Her bir değer, üçüncü bir isteğe bağlı parametresi bulunan kayıtlar arası fonksiyonun döndürdüğü tek bir değer ya da bir değer aralığı olabilir. Değerler serisi en az bir pozitif ve bir negatif değer içermelidir.
dates	Nakit akışı ödemelerine karşılık gelen bir ödeme tarihi veya ödeme tarihleri planı.

Sınırlamalar:

Metin değerleri, NULL değerler ve eksik değerler göz ardı edilir.

Tüm ödemelere 365 günlük yıl temel alınarak iskonto uygulanır.

Örnek tablo

Örnekler	Sonuçlar
`RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')`	80,25 döndürür

Örnek kodu uygulamanıza ekleyin ve çalıştırın. Sonucu görmek için, sonuçlar sütununda listelenen alanları uygulamanızda bir sayfaya ekleyin.
`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 '|');`
	Sonuçta oluşan tablo, tablodaki kayıtların her biri için döndürülen RangeXNPV değerlerini gösterir.
	RangeID — RangeXNPV
	1 — $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

5.22 Sıralama ve kümeleme işlevleri

Bu fonksiyonlar yalnızca grafik ifadelerinde kullanılabilir.

Grafiklerde fonksiyonları sıralama

Bu fonksiyonlar kullanıldığında, sıfır değerlerinin gösterilmemesi otomatik olarak devre dışı bırakılır. NULL değerler göz ardı edilir.

Rank

Rank(), ifadedeki grafiğin satırlarını değerlendirir ve her bir satır için, ifadede değerlendirilen boyutun değerinin göreli konumunu görüntüler. Fonksiyon ifadeyi değerlendirirken, sonucu, geçerli sütun segmentini içeren diğer satırların sonucuyla karşılaştırır ve geçerli satırın segment içindeki sıralamasını döndürür.

```
Rank - grafik fonksiyonu([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank(), ifadeyi değerlendirir ve sonucu, bir pivot tablonun geçerli satır segmentini içeren diğer sütunların sonucu ile karşılaştırır. Fonksiyon daha sonra, geçerli sütunun segment içindeki sıralamasını döndürür.

```
HRank - grafik fonksiyonu([TOTAL] expr[, mode[, fmt]])
```

Grafiklerdeki kümeleme işlevleri

KMeans2D

Site lisansı özellik grubu Qlik Sense sisteminin lisansı ile ilgili özellikleri içerir. Tüm alanlar zorunludur ve hiçbiri boş bırakılmamalıdır.

Site lisansı özellikleri

Özellik adı	Açıklama
Sahip adı	Qlik Sense ürünü sahibinin kullanıcı adı.
Sahip kuruluşu	Qlik Sense ürünü sahibinin üyesi olduğu kuruluşun adı.
Seri numarası	Qlik Sense yazılımına atanmış seri numarası.
Kontrol numarası	Qlik Sense yazılımına atanmış kontrol numarası.
LEF erişimi	Qlik Sense yazılımına atanmış License Enabler File (LEF).

KMeans2D(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin küme kimliğini görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar sırasıyla coordinate_1 ve coordinate_2 parametreleri tarafından belirlenir. Bunların her ikisi de toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir. Veriler isteğe bağlı olarak norm parametresi ile normalleştirilebilir.

```
KMeans2D - grafik fonksiyonu(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

KMeansND(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin küme kimliğini görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar, n sütuna kadar sırasıyla coordinate_1, coordinate_2 vb. parametreleri tarafından belirlenir. Bunların tümü toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir.

```
KMeansND - grafik fonksiyonu(num_clusters, num_iter, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin istenen koordinatını görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar sırasıyla coordinate_1 ve coordinate_2 parametreleri tarafından belirlenir. Bunların her ikisi de toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir. Veriler isteğe bağlı olarak norm parametresi ile normalleştirilebilir.

```
KMeansCentroid2D - grafik fonksiyonu(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin istenen koordinatını görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar, n sütuna kadar sırasıyla coordinate_1, coordinate_2 vb. parametreleri tarafından belirlenir. Bunların tümü toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir.

```
KMeansCentroidND - grafik fonksiyonu(num_clusters, num_iter, coordinate_no, coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Rank - grafik fonksiyonu

Rank(), ifadedeki grafiğin satırlarını değerlendirir ve her bir satır için, ifadede değerlendirilen boyutun değerinin göreli konumunu görüntüler. Fonksiyon ifadeyi değerlendirirken, sonucu, geçerli sütun segmentini içeren diğer satırların sonucuyla karşılaştırır ve geçerli satırın segment içindeki sıralamasını döndürür.

Sütun segmentleri

Tablolar dışındaki grafikler için, geçerli sütun segmenti grafiğin düz tablo eşdeğerinde görüldüğü gibi tanımlanır.

Söz Dizimi:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
mode	Fonksiyon sonucunun sayı temsilini belirtir.
fmt	Fonksiyon sonucunun metin temsilini belirtir.
TOTAL	Grafik tek boyutluysa veya ifadeden önce **TOTAL** niteleyicisi geliyorsa fonksiyon tüm sütun genelinde değerlendirilir. Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Sıralama ikili değer olarak döndürülür; bu, her satırın benzersiz bir sıralamaya sahip olduğu durumlarda 1 ile geçerli sütun segmentindeki satır sayısı arasında bir tamsayıdır.

Birkaç satırın aynı sıralamayı paylaştığı durumlarda, metin ve sayı temsili **mode** ve **fmt** parametreleriyle kontrol edilebilir.

mode
İkinci bağımsız değişken **mode** şu değerleri alabilir:

mode örnekleri

Değer	Açıklama
0 (varsayılan)	Paylaşma grubundaki tüm sıralamalar tüm sıralamanın orta değerinin düşük tarafına denk geliyorsa, tüm satırlar paylaşma grubu içindeki en düşük sıralamayı alır. Paylaşma grubundaki tüm sıralamalar tüm sıralamanın orta değerinin yüksek tarafına denk geliyorsa, tüm satırlar paylaşma grubu içindeki en yüksek sıralamayı alır. Paylaşma grubundaki sıralamalar tüm sıralamanın orta değeri üzerine yayılmışsa, tüm satırlar sütun segmentinin tamamındaki üst ve alt sıralamanın ortalamasına karşılık gelen değeri alır.
1	Tüm satırlarda en düşük sıralama.
2	Tüm satırlarda ortalama sıralama.
3	Tüm satırlarda en yüksek sıralama.
4	Birinci satırda en düşük sıralama, ardından her satır için bir birim artırılır.

fmt

Üçüncü bağımsız değişken **fmt** şu değerleri alabilir:

fmt örnekleri

Değer	Açıklama
0 (varsayılan)	Tüm satırlarda düşük değer - yüksek değer (örn. 3 - 4).
1	Tüm satırlarda düşük değer.
2	Birinci satırda düşük değer, sonraki satırlarda boş.

mode 4 ve **fmt** 2 için satırların sıralaması, grafik boyutlarının sıralama düzenine göre belirlenir.

Örnekler ve sonuçlar:

Product ile Sales boyutlarından bir grafik ve Product ile UnitSales boyutlarından bir diğer grafik olmak üzere iki görselleştirme oluşturun. Aşağıdaki tabloda gösterildiği gibi hesaplamaları ekleyin.

Sıralama örnekleri

Örnekler	Sonuçlar
Örnek 1. `Customer` ve `Sales` boyutları ve `Rank(Sales)` hesaplaması ile bir tablo oluşturun	Sonuç, boyutların sıralama düzenine göre değişir. Tablo Customer boyutuna göre sıralanırsa, tüm Sales değerleri (önce Astrida, sonra Betacab için vs.) tabloda listelenir. Rank (Sales) sonuçları Sales değeri 12 için 10, Sales değeri 13 için 9 vs. gösterir ve Sales değeri 78 için döndürülen sıralama değeri 1 olur. Bir sonraki sütun segmenti Betacab ile başlar ve bu öğe için segmentteki ilk Sales değeri 12'dir. Bunun için Rank(Sales) sıralama değeri 11 olarak verilmektedir. Tablo Sales öğesine göre sıralanırsa, sütun segmentleri Sales ve karşılık gelen Customer değerlerinden oluşur. İki Sales değeri 12 olduğundan (Astrida ve Betacab için), bu sütun segmenti için Rank(Sales) değeri 1-2'dir (her bir Customer değeri için). Bunun nedeni Sales değeri 12 için iki Customer değeri olmasıdır. 4 değer olsaydı, sonuç tüm satırlar için 1-4 olurdu. Bu, fmt bağımsız değişkeninin varsayılan değeri (0) için sonucun nasıl göründüğünü gösterir.
Örnek 2. Customer boyutunun yerine Product koyun ve `Rank(Sales,1,2)` hesaplamasını ekleyin	**mode** ve **fmt** bağımsız değişkenleri sırasıyla 1 ve 2 olarak ayarlandığından, bu ifade her bir sütun segmentinin ilk satırında 1 döndürür ve diğer satırları boş bırakır.

1. örneğin sonuçları (tablo Customer değerine göre sıralandığında):

Sonuçlar tablosu

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

1. örneğin sonuçları (tablo Sales değerine göre sıralandığında):

Sonuçlar tablosu

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

Örneklerde kullanılan veriler:

```
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 '|');
```

Ayrıca bkz.

Sum - grafik fonksiyonu (page 231)

HRank - grafik fonksiyonu

HRank(), ifadeyi değerlendirir ve sonucu, bir pivot tablonun geçerli satır segmentini içeren diğer sütunların sonucu ile karşılaştırır. Fonksiyon daha sonra, geçerli sütunun segment içindeki sıralamasını döndürür.

Söz Dizimi:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Dönüş verileri türü: dual

Bu fonksiyon yalnızca pivot tablolarda çalışır. Tüm diğer grafik türlerinde NULL döndürür.

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
expr	Hesaplanacak verileri içeren ifade veya alan.
mode	Fonksiyon sonucunun sayı temsilini belirtir.
fmt	Fonksiyon sonucunun metin temsilini belirtir.
TOTAL	Grafik tek boyutluysa veya ifadeden önce **TOTAL** niteleyicisi geliyorsa fonksiyon tüm sütun genelinde değerlendirilir. Tablo veya tablo eşdeğeri birden çok dikey boyuta sahipse, geçerli sütun segmenti, alanlar arası sıralama düzenindeki son boyutu gösteren sütun haricinde tüm boyut sütunlarında geçerli satır olarak yalnızca aynı değerlere sahip satırları içerir.

Pivot tablo tek boyutluysa veya ifadeden önce **total** niteleyicisi geliyorsa, geçerli satır segmenti her zaman satırın tamamına eşittir. Pivot tablo birden çok yatay boyuta sahipse, geçerli satır segmenti, alanlar arası sıralama düzeninin son yatay boyutunu gösteren satır haricinde tüm boyut satırlarında geçerli sütun olarak yalnızca aynı değerlere sahip sütunları içerir.

Sıralama ikili değer olarak döndürülür; bu, her sütunun benzersiz bir sıralamaya sahip olduğu durumlarda 1 ile geçerli satır segmentindeki sütun sayısı arasında bir tamsayıdır.

Birkaç sütunun aynı sıralamayı paylaştığı durumlarda, metin ve sayı temsili **mode** ve **format** bağımsız değişkenleriyle kontrol edilebilir.

İkinci bağımsız değişken (**mode**), fonksiyon sonucunun sayı temsilini belirtir:

mode örnekleri

Değer	Açıklama
0 (varsayılan)	Paylaşma grubundaki tüm sıralamalar tüm sıralamanın orta değerinin düşük tarafına denk geliyorsa, tüm sütunlar paylaşma grubu içindeki en düşük sıralamayı alır. Paylaşma grubundaki tüm sıralamalar tüm sıralamanın orta değerinin yüksek tarafına denk geliyorsa, tüm sütunlar paylaşma grubu içindeki en yüksek sıralamayı alır. Paylaşma grubundaki sıralamalar tüm sıralamanın orta değeri üzerine yayılmışsa, tüm satırlar sütun segmentinin tamamındaki üst ve alt sıralamanın ortalamasına karşılık gelen değeri alır.
1	Gruptaki tüm sütunlarda en düşük sıralama.
2	Gruptaki tüm sütunlarda ortalama sıralama.

Değer	Açıklama
3	Gruptaki tüm sütunlarda en yüksek sıralama.
4	Birinci sütunda en düşük sıralama, ardından gruptaki her sütun için bir birim artırılır.

Üçüncü bağımsız değişken (**format**), fonksiyon sonucunun metin temsilini belirtir:

format örnekleri

Değer	Açıklama
0 (varsayılan)	Gruptaki tüm sütunlarda düşük değer**&' - '&**yüksek değer (örn. 3 - 4).
1	Gruptaki tüm sütunlarda düşük değer.
2	Birinci sütunda düşük değer, gruptaki sonraki sütunlarda boş.

mode 4 ve **format** 2 için sütunların sıralaması, grafik boyutlarının sıralama düzenine göre belirlenir.

Örnekler:

```
HRank( sum( Sales ))
HRank( sum( Sales ), 2 )
HRank( sum( Sales ), 0, 1 )
```

K-ortalamaları ile optimizasyon: Gerçek dünyadan bir örnek

Aşağıdaki örnek, K-Ortalamaları kümeleme ve Sentroid işlevlerinin bir veri kümesine uygulandığı, gerçek dünyadan bir örnektir. K-Ortalamaları fonksiyonu, veri noktalarını benzerlikleri olan kümeler halinde ayırır. K-Ortalamaları algoritması yapılandırılabilir sayıda yineleme üzerine uygulandığından, kümeler daha kompakt ve farklı hale gelir.

K-Ortalamaları, çok çeşitli kullanım durumlarında birçok alanda kullanılır; kümeleme kullanım senaryolarıyla ilgili bazı örnekler arasında müşteri segmentasyonu, dolandırıcılık tespiti, hesap yıpranmasını tahmin etme, müşteri teşviklerini hedefleme, siber suçluları tanımlama ve teslimat rotası optimizasyonu yer alır. K-Ortalamaları kümeleme algoritması, işletmelerin kalıpları belirlemeye ve hizmet tekliflerini optimize etmeye çalıştığı durumlarda giderek daha fazla kullanılmaktadır.

Qlik Sense K-Ortalamaları ve Sentroid işlevleri

Qlik Sense, veri noktalarını benzerliğe dayalı olarak kümeler halinde gruplayan iki K-Ortalamaları fonksiyonu sağlar. Bkz. *KMeans2D - grafik fonksiyonu (page 756)* ve *KMeansND - grafik fonksiyonu (page 767)*. **KMeans2D** işlevi iki boyut kabul eder ve sonuçları bir **dağılım grafiği** aracılığıyla görselleştirirken çok işe yarar. **KMeansND** işlevi ikiden fazla boyut kabul eder. Standart grafiklerde bir 2B sonucu kavramsallaştırmak kolay olduğu için, aşağıdaki gösterim iki boyut kullanarak bir **dağılım grafiği** üzerinde K-Ortalamaları'nı uygulayacaktır. K-Ortalamaları kümelemesi, ifadeye göre renklendirme yoluyla veya bu örnekte açıklandığı gibi boyuta göre görselleştirilebilir.

Qlik Sense sentroid işlevi, kümedeki tüm veri noktalarının aritmetik ortalama konumunu belirler ve bu küme için bir merkezi nokta veya sentroid tanımlar. Her bir grafik satırı (veya kaydı) için, sentroid işlevi bu veri noktasının atandığı kümenin koordinatını görüntüler. Bkz. *KMeansCentroid2D - grafik fonksiyonu (page 778)* ve *KMeansCentroidND - grafik fonksiyonu (page 780)*.

Kullanım senaryosu ve örneğe genel bakış

Aşağıdaki örnek, simüle edilmiş bir gerçek dünya senaryosuyla ilgilidir. ABD'nin New York eyaletindeki bir tekstil firması, teslimat maliyetlerini en aza indirerek giderlerini azaltmak istiyor. Bunu yapmanın bir yolu, depoların yerini distribütörlerine daha yakın olacak şekilde değiştirmektir. Şirketin, New York eyaleti genelinde 118 distribütörü vardır. Aşağıdaki gösterim, bir operasyon müdürünün K-Ortalamaları fonksiyonunu kullanarak distribütörleri benzer şekilde kümelenmiş beş coğrafyaya nasıl bölebileceğine ve ardından sentroid işlevini kullanarak bu kümelerin merkezindeki beş ideal depo konumunu nasıl belirleyebileceğine ilişkin bir simülasyondur. Amaç, beş merkezi depo yerini belirlemek için kullanılabilecek eşleme koordinatlarını keşfetmektir.

Veri kümesi

Veri kümesi, gerçek enlem ve boylam koordinatlarıyla New York eyaletinde rastgele oluşturulmuş adlara ve adreslere dayanmaktadır. Veri kümesi şu on sütunu içerir: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Veri kümesi, yerel olarak indirebileceğiniz ve ardından Qlik Sense uygulamasına veya satır içi olarak veri yükleme düzenleyicisine yükleyebileceğiniz bir dosya olarak aşağıda mevcuttur. Oluşturulan uygulama *Distribütörler K-Ortalamaları ve Sentroid* olarak, uygulamadaki ilk sayfa ise *Dağıtım küme analizi* olarak adlandırılır.

Örnek veri dosyasını indirmek için aşağıdaki bağlantıyı seçin: DistributorData.csv

Distributor veri kümesi: Qlik Sense içinde veri yükleme düzenleyicisi için satır içi yükleme (page 754)

Başlık: DistributorData

Toplam kayıt sayısı: 118

KMeans2D işlevini uygulama

Bu örnekte, bir **dağılım grafiğinin** yapılandırması *DistributorData* veri kümesi kullanılarak gösterilir, **KMeans2D** işlevi uygulanır ve grafik boyuta göre renklendirilir.

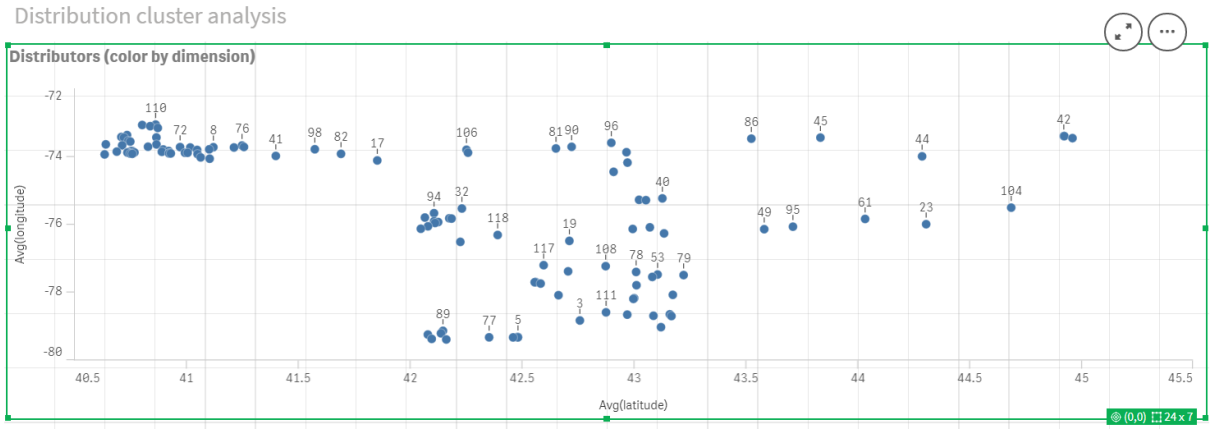
Qlik Sense K-Ortalamaları fonksiyonunun, derinlik farkı (DeD) adlı bir yöntem kullanılarak otomatik kümelemeyi desteklediğine dikkat edin. Bir kullanıcı, küme sayısı için 0 değerini ayarladığında, o veri kümesi için ideal küme sayısı belirlenir. Ancak bu örnekte, **num_clusters** bağımsız değişkeni için bir değişken oluşturulmuştur (söz dizimi için bkz. *KMeans2D - grafik fonksiyonu (page 756)*). Bu nedenle istenen küme sayısı (k=5) bir değişken tarafından belirlenir.

1. Bir **dağılım grafiği** sayfaya sürüklenir ve *Distribütörler (boyuta göre)* olarak adlandırılır.
2. Küme sayısını belirtmek için bir **değişken** oluşturulur. **Değişkenin** adı *vDistClusters*'dır. **Tanım** değişkeni için *5* girilir.
3. Grafiğin **veri** yapılandırması:
    a. **Boyutlar** altında, **Kabarcık** için *kimlik* alanı seçilir. *Etiket* için **Küme kimliği** girilir.
    b. **Hesaplamalar** altında, **X ekseni**'nin ifadesi *Avg([latitude])*'dir.

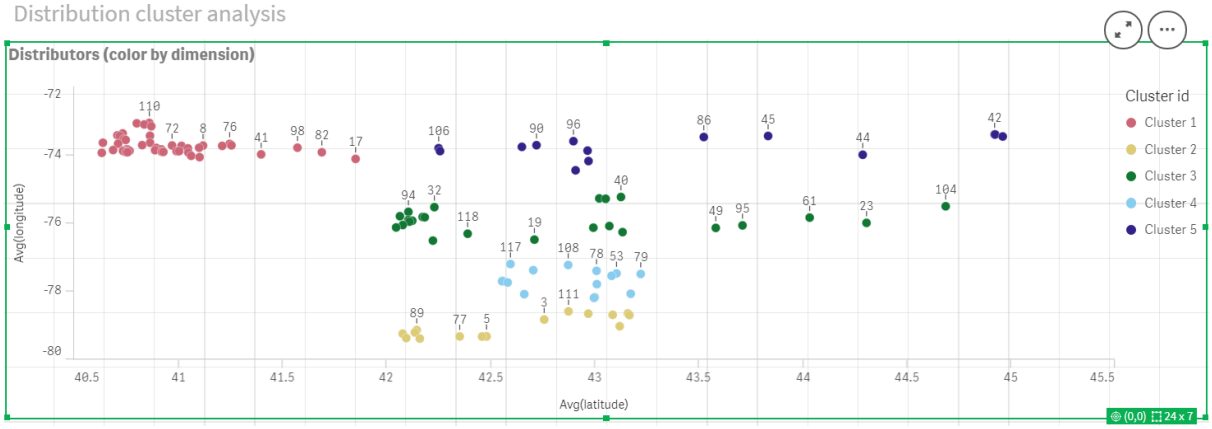
c. **Hesaplamalar** altında, *Y ekseninin* ifadesi **Avg([longitude])**'dur.

4. **Görünüş** yapılandırması:
   a. **Renkler ve gösterge** altında, **Renkler** için **Özel** seçilir.
   b. Grafiği renklendirmek için **Boyuta göre** seçilir.
   c. Şu ifade girilir: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Küme 1', 'Küme 2', 'Küme 3', 'Küme 4', 'Küme 5')*
   d. **Kalıcı renkler** onay kutusu seçilir.

Boyuta göre K-Ortalamaları renklendirme uygulanmadan önce dağılım grafiği



Boyuta göre K-Ortalamaları renklendirme uygulandıktan sonra dağılım grafiği



Tablo ekleme: *Distribütörler*

İlgili verilere hızlı erişim için bir tablonun hazır bulundurulması yararlı olabilir. **Dağılım grafiği** referans için ilgili distribütör adlarına sahip bir tablo eklenmesine rağmen *kimlikleri* gösterir.

1. *Distribütörler* adlı bir **tablo**, şu **Sütunlar** (Boyutlar) eklenmiş olarak sayfaya sürüklenir: *id*, *first_name* ve *last_name*.

Tablo: Distribütör adları

Distributors

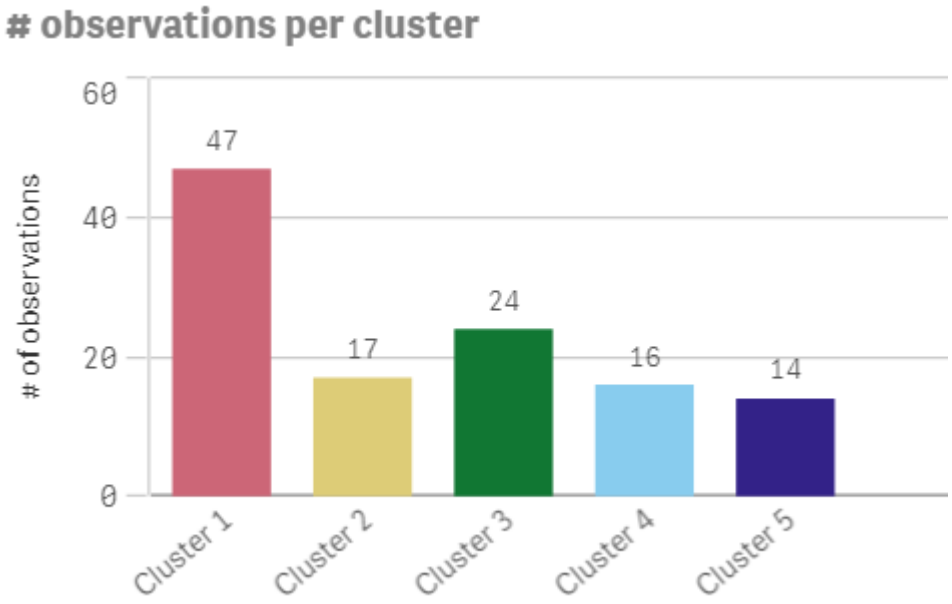
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Çubuk grafik ekleme: *küme başına gözlem sayısı*

Depo dağıtım senaryosu için, her bir depo tarafından kaç distribütöre hizmet verileceğini bilmek yararlı olur. Bu nedenle, her kümeye kaç distribütörün atandığını hesaplayan bir **çubuk grafik** oluşturulur.

1. Sayfaya bir **çubuk grafik** sürüklenir. Grafiğin adı: *küme başına gözlem sayısı*.
2. **Çubuk grafik** için **veri** yapılandırması:
    a. *Kümeler* etiketli bir **Boyut** eklenir (etiket, ifade uygulandıktan sonra eklenebilir). Şu ifade girilir: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Küme 1', 'Küme 2', 'Küme 3', 'Küme 4', 'Küme 5')*
    b. *Gözlem sayısı* etiketli bir **Hesaplama** eklenir. Şu ifade girilir: *=count(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id))*
3. **Görünüş** yapılandırması:
    a. **Renkler ve gösterge** altında, **Renkler** için **Özel** seçilir.
    b. Grafiği renklendirmek için **Boyuta göre** seçilir.
    c. Şu ifade girilir: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Küme 1', 'Küme 2', 'Küme 3', 'Küme 4', 'Küme 5')*
    d. **Kalıcı renkler** onay kutusu seçilir.
    e. **Göstergeyi göster** kapalıdır.
    f. **Sunum** altındaki **Değer etiketleri**, **Otomatik** seçeneğine ayarlanır.
    g. **X ekseni** altında: **Kümeler**, **Yalnızca etiketler** seçilir.

Çubuk grafik: Küme başına gözlem sayısı



Centroid2D fonksiyonunu uygulama

Centroid2D fonksiyonu için, olası depo konumlarının koordinatlarını belirleyecek olan ikinci bir tablo eklenir. Bu tablo, tanımlanan beş distribütör grubu için merkezi konumu (merkez değerleri) gösterir.

1. Sayfaya bir **Tablo** sürüklenir, *Küme merkezleri* olarak adlandırılır ve aşağıdaki sütunlar eklenir:
    a. **Kümeler** etiketli bir *Boyut* eklenir. Şu ifade girilir: *=pick(aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)+1,'Depo 1','Depo 2','Depo 3','Depo 4','Depo 5'*)
    b. **enlem (D1)** etiketli bir *Hesaplama* eklenir. Şu ifade girilir: *=only(aggr(KMeansCentroid2D (vDistClusters,0,only(latitude),only(longitude)),id))*
    coordinate_no parametresinin birinci boyuta(0) karşılık geldiğine dikkat edin. Bu durumda *enlem* boyutu, x eksenine çizilir. **CentroidND** fonksiyonuyla çalışıyor olsaydık ve en fazla altı boyut olsaydı bu parametre girişleri altı değerden (0, 1, 2, 3, 4 veya 5) herhangi biri olabilirdi.
    c. *Enlem (D2)* etiketli **Hesaplama** eklenir. Şu ifade girilir: *=only(aggr(KMeansCentroid2D (vDistClusters,1,only(latitude),only(longitude)),id))*
    Bu ifadedeki **coordinate_no** parametresi ikinci boyuta(1) karşılık gelir. *Boylam* boyutu y eksenine çizilir.

Tablo: Küme sentroid hesaplamaları

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

Sentroid haritalama

Sıradaki adım sentroidleri haritalamaktır. Görselleştirmeyi ayrı sayfalara yerleştirmeyi tercih edip etmemek uygulama geliştiricisine bağlıdır.

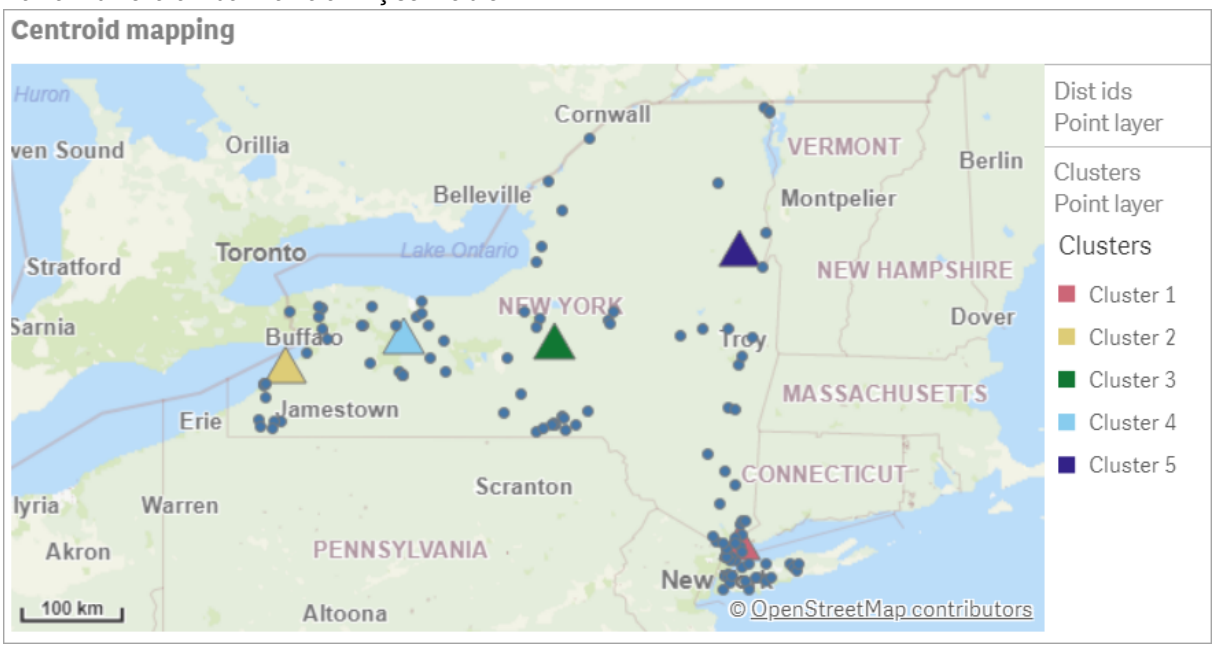
1. *Sentroid haritalama* adlı bir **harita** sayfaya sürüklenir.
2. **Katmanlar** bölümünde. **Katman ekle** seçilir, ardından **Nokta katmanı** seçilir.
    a. **Alan** *kimliği* seçilir ve *Distribütör kimlikleri* **Etiketi** eklenir.
    b. **Konum** bölümünde **Enlem ve Boylam alanları** için onay kutusu seçilir.
    c. **Enlem** için *enlem* alanı seçilir.
    d. **Boylam** için *boylam* alanı seçilir.
    e. **Boyut ve Şekil** bölümünde, **Şekil** için **Kabarcık** seçilir ve **Boyut** kaydırıcıda istenen tercihe indirgenir.
    f. **Renkler** bölümünde **Tek renk** seçilir, **Renk** için mavi, **Anahat** rengi için gri seçilir (bu seçimler de tercih meselesidir).
3. **Katmanlar** bölümünde, **Katman ekle** ve ardından **Nokta katmanı** seçilerek ikinci bir **Nokta katmanı** eklenir.
    a. Şu ifade girilir: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
    b. **Etiket** olarak *Kümeler* eklenir.
    c. **Konum** bölümünde **Enlem ve Boylam alanları** için onay kutusu seçilir.
    d. Bu örnekte x eksenine çizilen **Enlem** için şu ifade eklenir: *=aggr(KMeansCentroid2D (vDistClusters,0,only(latitude),only(longitude)),id)*
    e. Bu örnekte y eksenine çizilen **Boylam** için şu ifade eklenir: *=aggr(KMeansCentroid2D (vDistClusters,1,only(latitude),only(longitude)),id)*
    f. **Boyut ve Şekil** bölümünde, **Şekil** için **Üçgen** seçilir, **Boyut** ise kaydırıcıda istenen tercihe ayarlanarak küçültülür.
    g. **Renkler ve gösterge** altında, **Renkler** için **Özel** seçilir.

h. Grafiği renklendirmek için **Boyuta göre** seçilir. Şu ifade girilir: *=pick(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id)+1, 'Küme 1', 'Küme 2', 'Küme 3', 'Küme 4', 'Küme 5')*

i. Boyut, *Kümeler* olarak etiketlenir.

4. **Harita ayarlarında**, **Projeksiyon** için **Uyarlamalı** seçilir. **Hesaplama birimi** için **Metrik** seçilir.

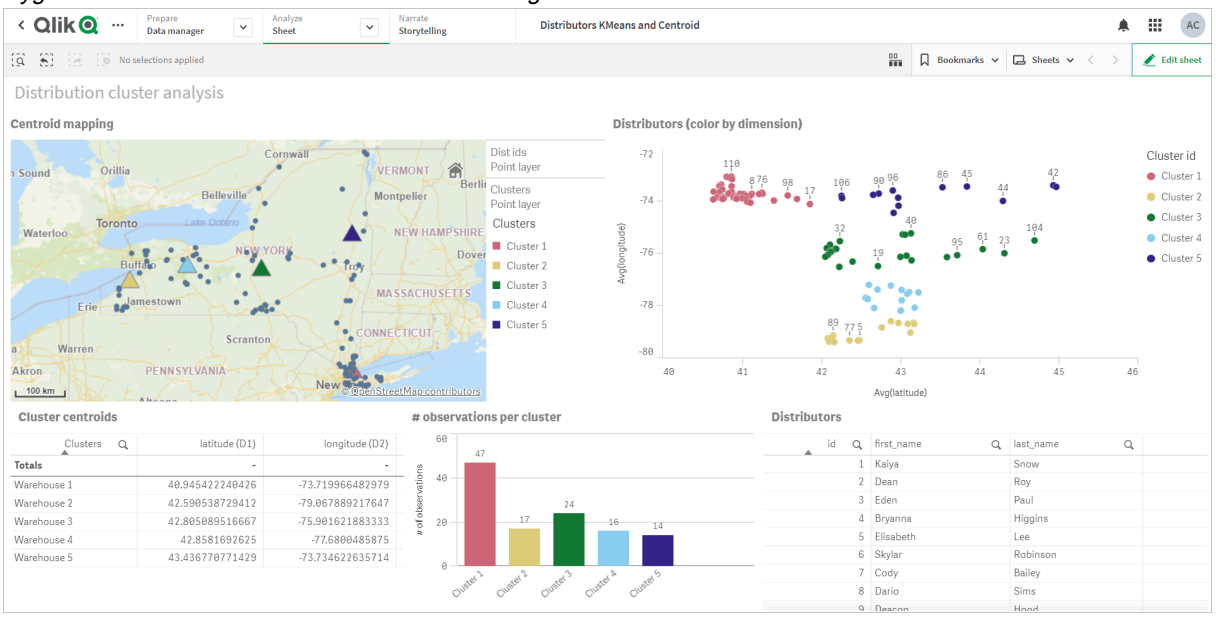
Harita: Küme tarafından haritalanmış sentroidler



Sonuç

Bu gerçek dünya senaryosu için K-Ortalamaları fonksiyonunu kullanarak, distribütörler benzerliğe (bu örnekte, birbirine yakınlık) dayalı olarak benzer gruplara veya kümelere ayrılmıştır. Beş harita koordinatını belirlemek için bu kümelere Sentroid fonksiyonu uygulandı. Bu koordinatlar, depoların inşa edileceği veya yerleştirileceği bir ilk merkezi konum sağlar. Sentroid fonksiyonu **harita** grafiğine uygulanır, böylece uygulama kullanıcıları, çevredeki küme veri noktalarına göre merkez noktalarının nerede bulunduğunu görselleştirebilir. Elde edilen koordinatlar New York eyaletindeki distribütörlere teslimat maliyetlerini en aza indirebilecek olası depo konumlarını temsil eder.

Uygulama: K-Ortalamaları ve sentroid analizi örneği



Distributor veri kümesi: Qlik Sense içinde veri yükleme düzenleyicisi için satır içi yükleme

```
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 - grafik fonksiyonu

KMeans2D(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin küme kimliğini görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar sırasıyla coordinate_1 ve coordinate_2 parametreleri tarafından belirlenir. Bunların her ikisi de toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir. Veriler isteğe bağlı olarak norm parametresi ile normalleştirilebilir.

KMeans2D, veri noktası başına tek bir değer döndürür. Döndürülen değer, ikili değerdir ve her bir veri noktasının atanmış olduğu kümeye karşılık gelen tamsayı değeridir.

Söz Dizimi:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
num_clusters	Küme sayısını belirten tamsayı.
coordinate_1	Birinci koordinatı (genellikle grafikten oluşturulabilen dağılım grafiğinin x ekseni) hesaplayan toplama. coordinate_2 adlı ek parametre, ikinci koordinatı hesaplar.
norm	KMeans kümelemesinden önce veri kümelerine uygulanan isteğe bağlı normalleştirme yöntemi. Olası değerler: Normalleştirme yok için 0 veya "none" z puanı normalleştirme için "zscore" Min-maks normalleştirme için 2 veya "minmax" Hiç parametre sağlanmadıysa veya sağlanan parametre yanlışsa, normalleştirme yok uygulanır. Z puanı, özellik ortalamasına ve standart sapmaya göre verileri normalleştirir. Z puanı, her özelliğin aynı ölçeğe sahip olmasını sağlamaz, ancak aykırı değerlerle çalışırken min-maks'tan daha iyi bir yaklaşımdır. Min-maks normalleştirme, her birinin minimum ve maksimum değerlerini alarak ve her bir veri noktasını yeniden hesaplayarak özelliklerin aynı ölçeğe sahip olmasını sağlar.

Örnek: Grafik ifadesi
Bu örnekte, *Iris* veri kümesini kullanarak bir dağılım çizimi grafiği oluşturup verileri ifadeye göre renklendirmek için KMeans kullanırız.

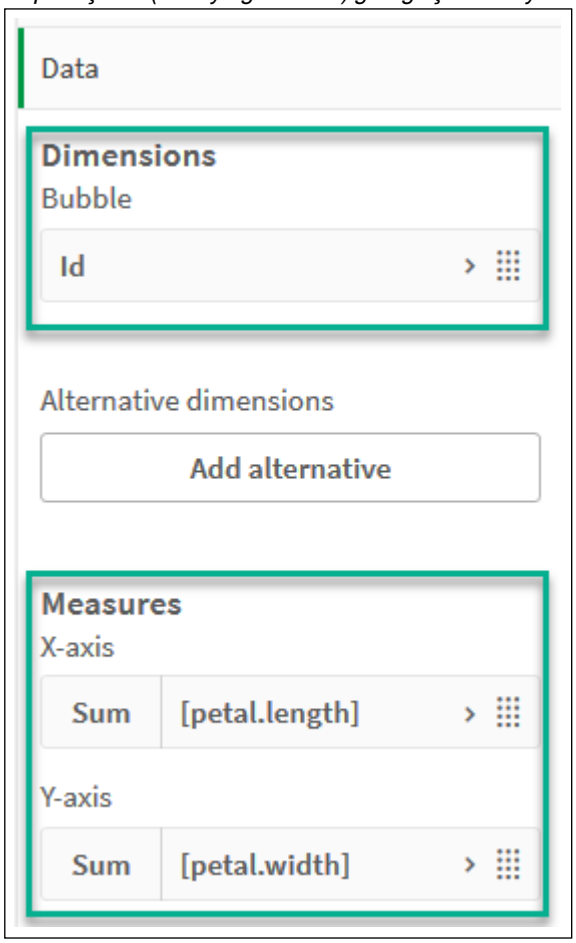
Ayrıca *num_clusters* bağımsız değişkeni için bir değişken de oluşturur ve sonra küme sayısını değiştirmek için bir değişken giriş kutusu kullanırız.

Iris veri kümesi çeşitli biçimlerde genel kullanıma sunulur. Qlik Sense içinde veri yükleme düzenleyicisini kullanarak verileri yüklenecek satır içi tablo olarak sağladık. Bu örnek için veri tablosuna bir *Kimlik* sütunu eklediğimizi unutmayın.

Qlik Sense uygulamasına verileri yükledikten sonra şunları yaparız:

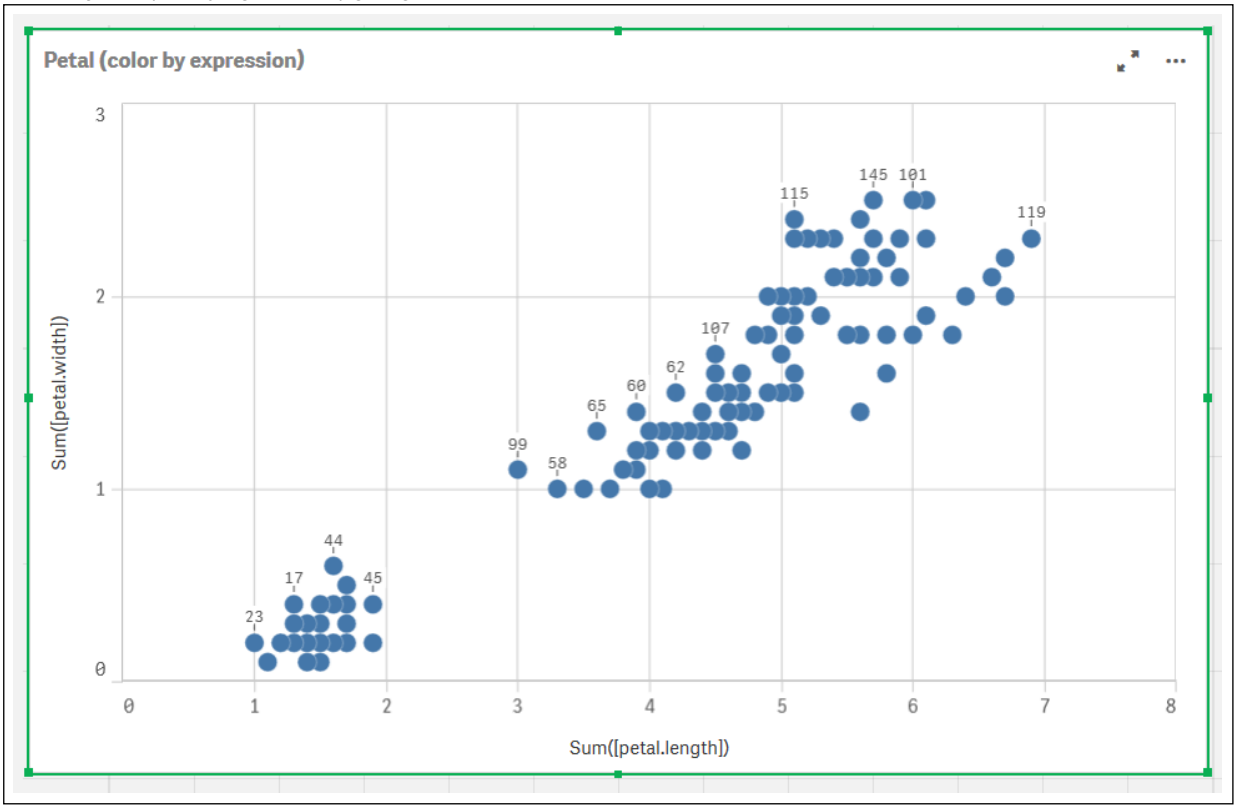
1. Bir **Dağılım çizimi** grafiğini yeni bir sayfaya sürükleyin. Grafiği *Yaprak şema (ifadeye göre renk)* olarak adlandırın.
2. Küme sayısını belirtmek için bir değişken oluşturun. **Ad** değişkeni için *KmeansPetalClusters* girin. **Tanım** değişkeni için *=2* girin.
3. Grafik için **Veri**'yi yapılandırma:
    i. **Boyutlar** bölümünde, **Kabarcık** için alanın *kimlik* bilgisini seçin. Etiket için Küme Kimliğini girin.
    ii. **Hesaplamalar** bölümünde, **X ekseni** ifadesi için *Sum([petal.length])* seçeneğini belirleyin.
    iii. **Hesaplamalar** bölümünde, **Y ekseni** ifadesi için *Sum([petal.width])* seçeneğini belirleyin.

    Yaprak şema (ifadeye göre renk) grafiği için veri ayarları

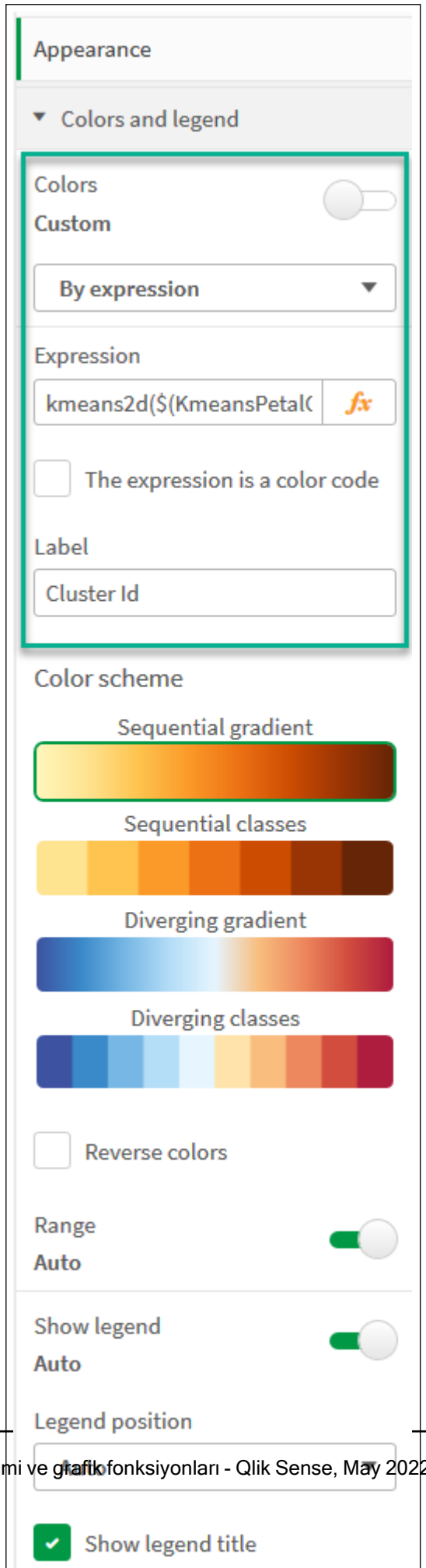
    Veri noktaları, grafikte çizilir.

Yaprak şema (ifadeye göre renk) grafiğindeki veri noktaları



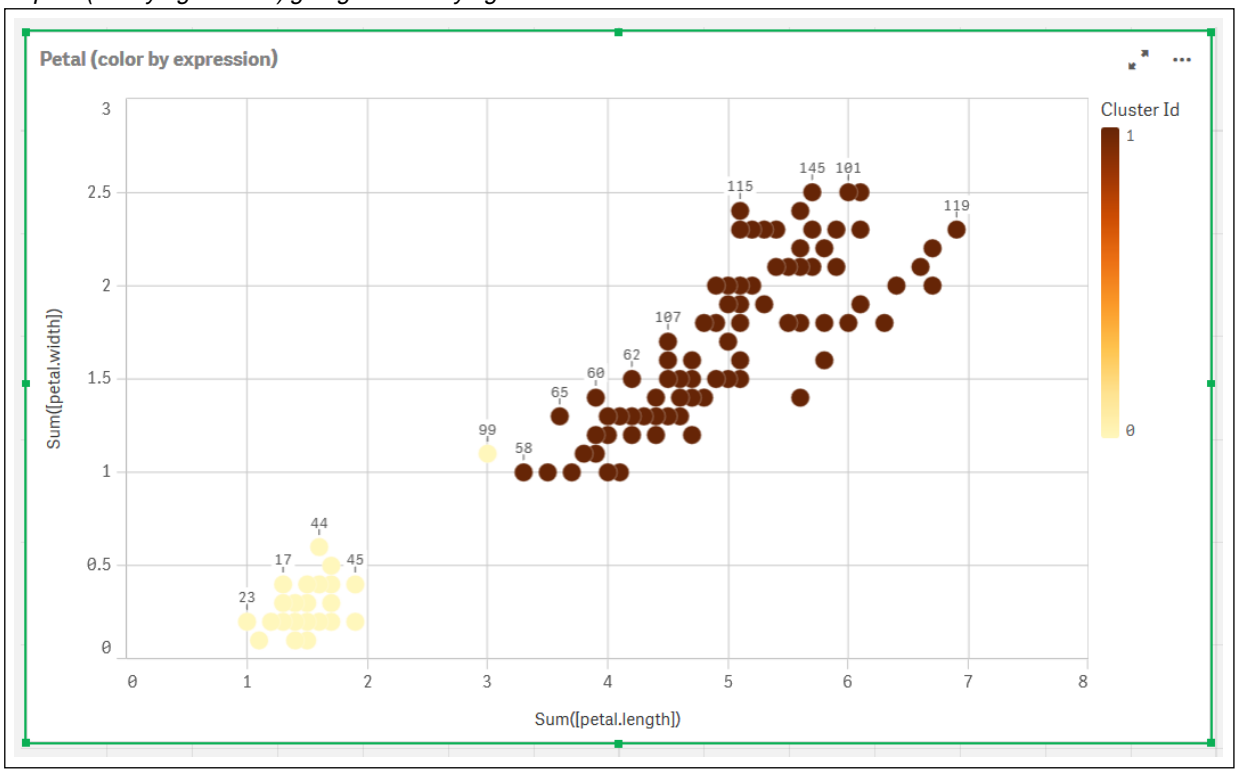
4. Grafik için **Görünüş**'ü yapılandırma:
    i. **Renkler ve gösterge** bölümünde, **Renkler** için **Özel**'i seçin.
    ii. Grafiği **İfadeye göre** renklendirmek için seçin.
    iii. **İfade** için şunu girin: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum([petal.width]))*
    KmeansPetalClusters öğesinin, *2* olarak ayarladığımız değişken olduğunu unutmayın.
    Alternatif olarak şunu girin: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. **İfade bir renk kodudur** onay kutusunun seçimini kaldırın.
    v. **Etiket** için şunu girin: *Küme Kimliği*

Yaprak (ifadeye göre renk) grafiğinin görünüş ayarları


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Show legend title

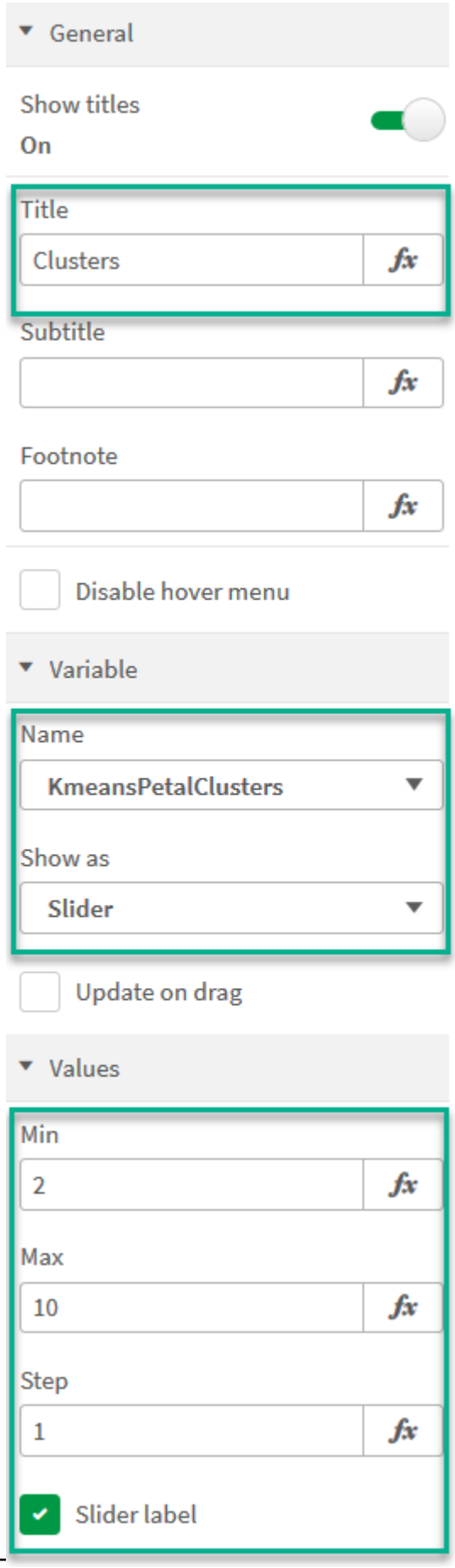
Grafikteki iki küme, KMeans ifadesine göre renklendirilir.

Yaprak (ifadeye göre renk) grafiğinde ifadeye göre renklendirilen kümeler



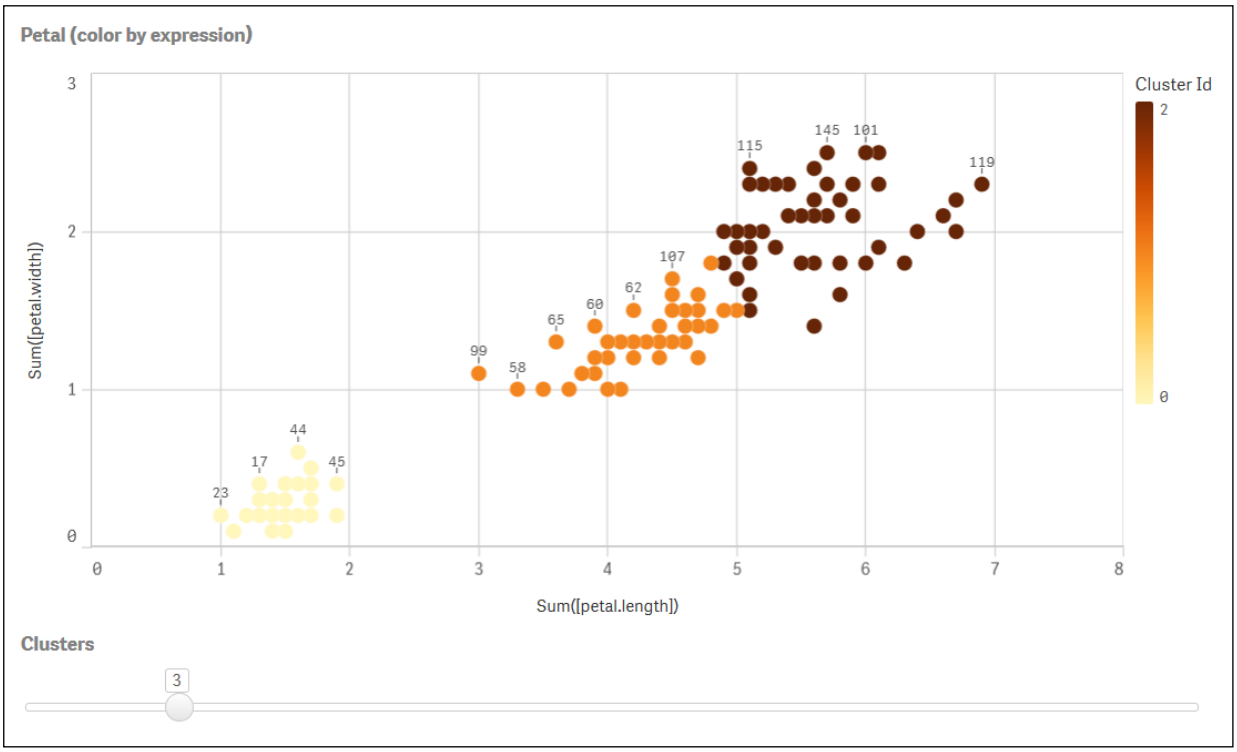
5. Küme sayısı için bir **Değişken girişi** kutusu ekleyin.
    i. **Varlıklar** panelindeki **Özel nesneler** bölümünde **Qlik Gösterge Paneli paketi**'ni seçin. Gösterge paneli paketine erişimimiz olmasaydı da oluşturduğumuz değişkeni kullanarak veya ifadede tamsayı olarak doğrudan küme sayısını değiştirebilirdik.
    ii. Bir **Değişken girişi** kutusunu sayfaya sürükleyin.
    iii. **Görünüm** bölümünde **Genel**'e tıklayın.
    iv. **Başlık** için şunu girin: *Kümeler*
    v. **Değişken**'e tıklayın.
    vi. **Ad** için şu değişkeni seçin: *KmeansPetalClusters*.
    vii. **Farklı göster** için **Kaydırıcı**'yı seçin.
    viii. **Değerler**'i seçin ve ayarları gerektiği şekilde yapılandırın.

Kümeler değişkeni giriş kutusu görünümü


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

Düzenlememiz bittiğinde, *Kümeler* değişken giriş kutusundaki kaydırıcıyı kullanarak küme sayısını değiştirebiliriz.

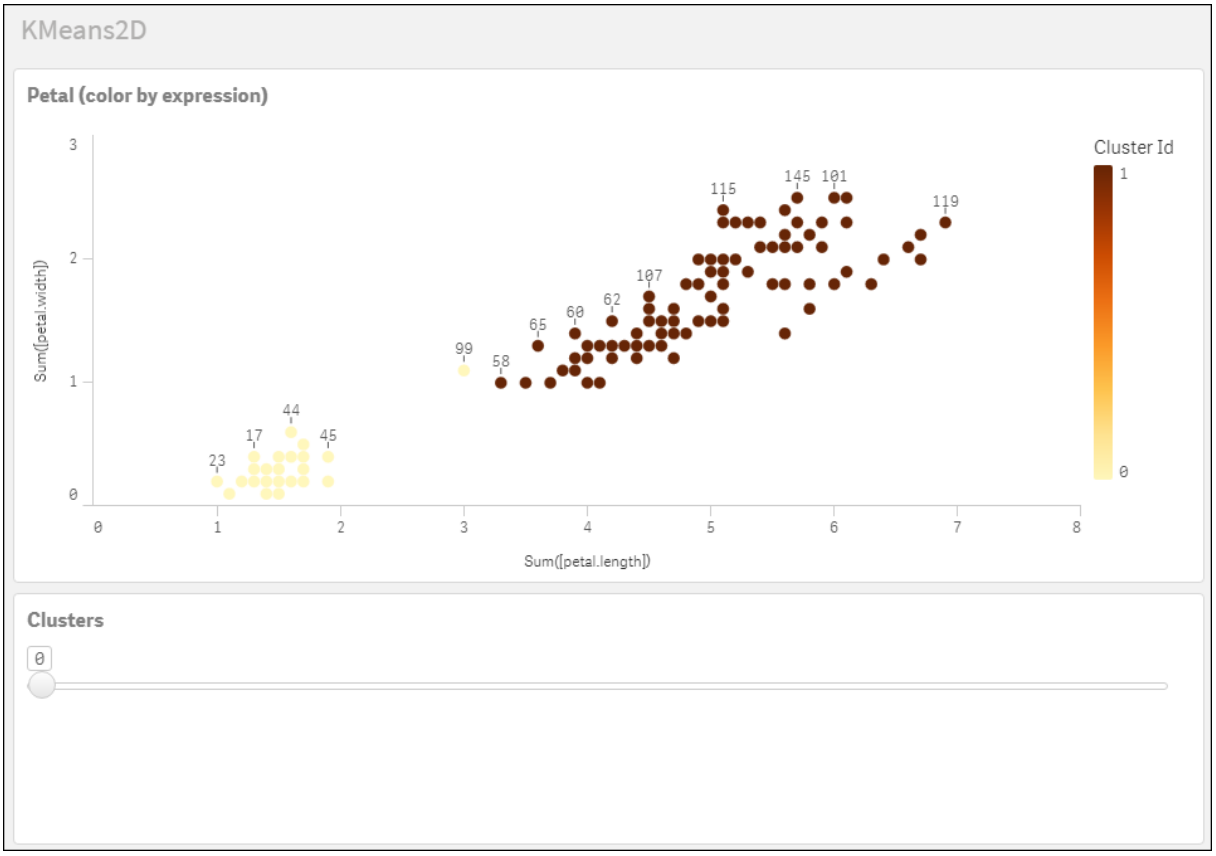
Yaprak (ifadeye göre renk) grafiğinde ifadeye göre renklendirilen kümeler



Otomatik kümeleme

KMeans işlevleri, derinlik farkı (DeD) adlı bir yöntem kullanılarak otomatik kümelemeyi destekler. Bir kullanıcı, küme sayısı için 0 değerini ayarladığında, o veri kümesi için optimum küme sayısı belirlenir. Küme sayısı (*k*) için bir tamsayı, belirtik şekilde döndürülmese de KMeans algoritması içinde hesaplanır. Örneğin, *KmeansPetalClusters* değeri için işlevde 0 değeri belirtilirse veya bir değişken giriş kutusu aracılığıyla ayarlanırsa optimum bir küme sayısına dayalı olarak veri kümesi için küme atamaları otomatik şekilde hesaplanır.

K-Ortalamaları derinlik farkı yöntemi, (k) değeri 0 olarak ayarlandığı andaki optimum küme sayısını belirler



Iris veri kümesi: Qlik Sense içinde veri yükleme düzenleyicisi için satır içi yükleme

```
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 - grafik fonksiyonu

KMeansND(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin küme kimliğini görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar, n sütuna kadar sırasıyla coordinate_1, coordinate_2 vb. parametreleri tarafından belirlenir. Bunların tümü toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir.

KMeansND, veri noktası başına tek bir değer döndürür. Döndürülen değer, ikili değerdir ve her bir veri noktasının atanmış olduğu kümeye karşılık gelen tamsayı değeridir.

Söz Dizimi:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
num_clusters	Küme sayısını belirten tamsayı.
num_iter	Yeniden başlatılmış küme merkezleri ile kümeleme yinelemesi sayısı.
coordinate_1	Birinci koordinatı (genellikle grafikten oluşturulabilen dağılım grafiğinin x ekseni) hesaplayan toplama. Ek parametreler ikinci, üçüncü ve dördüncü koordinatları vb. hesaplar.

Örnek: Grafik ifadesi
Bu örnekte, *Iris* veri kümesini kullanarak bir dağılım çizimi grafiği oluşturup verileri ifadeye göre renklendirmek için KMeans kullanırız.

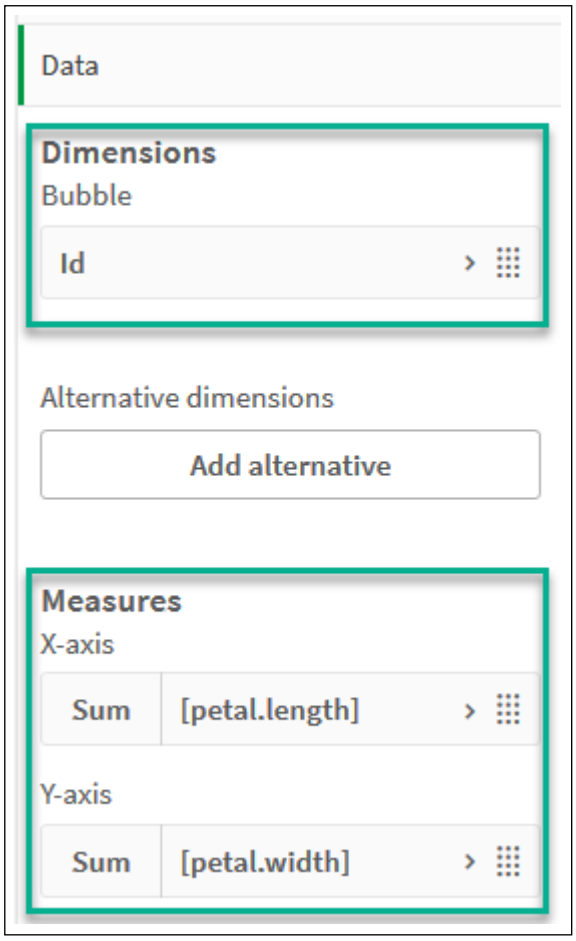
Ayrıca *num_clusters* bağımsız değişkeni için bir değişken de oluşturur ve sonra küme sayısını değiştirmek için bir değişken giriş kutusu kullanırız.

Ek olarak, *num_iter* bağımsız değişkeni için bir değişken oluşturur ve sonra yineleme sayısını değiştirmek için ikinci bir değişken giriş kutusu kullanırız.

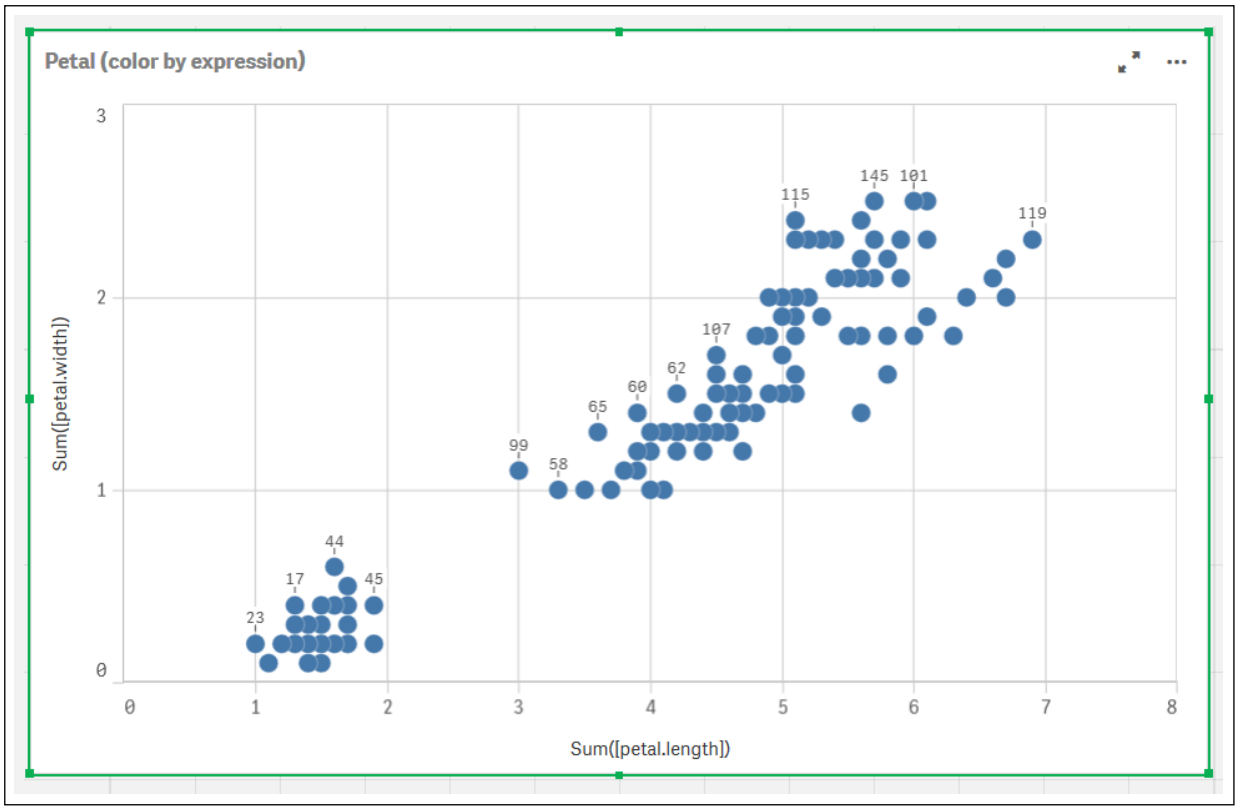
Iris veri kümesi çeşitli biçimlerde genel kullanıma sunulur. Qlik Sense içinde veri yükleme düzenleyicisini kullanarak verileri yüklenecek satır içi tablo olarak sağladık. Bu örnek için veri tablosuna bir *Kimlik* sütunu eklediğimizi unutmayın.

Qlik Sense uygulamasına verileri yükledikten sonra şunları yaparız:

1. Bir **Dağılım çizimi** grafiğini yeni bir sayfaya sürükleyin. Grafiği *Yaprak şema (ifadeye göre renk)* olarak adlandırın.
2. Küme sayısını belirtmek için bir değişken oluşturun. **Ad** değişkeni için *KmeansPetalClusters* girin. **Tanım** değişkeni için *=2* girin.
3. Yineleme sayısını belirtmek için bir değişken oluşturun. **Ad** değişkeni için *KmeansNumberIterations* girin. **Tanım** değişkeni için *=1* girin.
4. Grafik için **Veri**'yi yapılandırma:
    i. **Boyutlar** bölümünde, **Kabarcık** için alanın *kimlik* bilgisini seçin. Etiket için Küme Kimliğini girin.
    ii. **Hesaplamalar** bölümünde, **X ekseni** ifadesi için *Sum([petal.length])* seçeneğini belirleyin.
    iii. **Hesaplamalar** bölümünde, **Y ekseni** ifadesi için *Sum([petal.width])* seçeneğini belirleyin.
    Yaprak şema (ifadeye göre renk) grafiği için veri ayarları

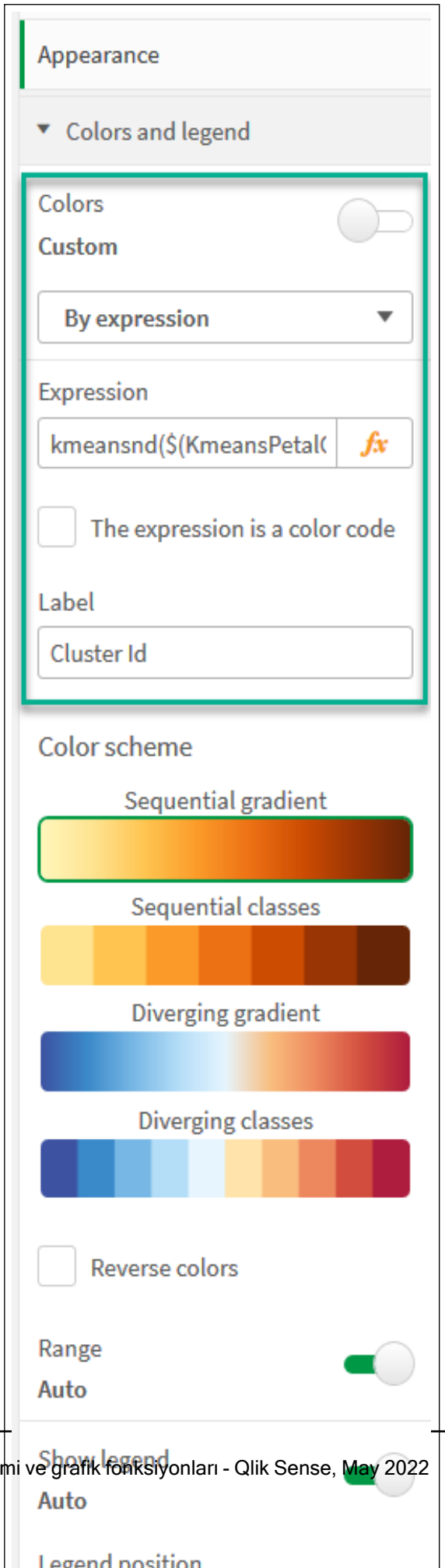


Veri noktaları, grafikte çizilir.
Yaprak şema (ifadeye göre renk) grafiğindeki veri noktaları



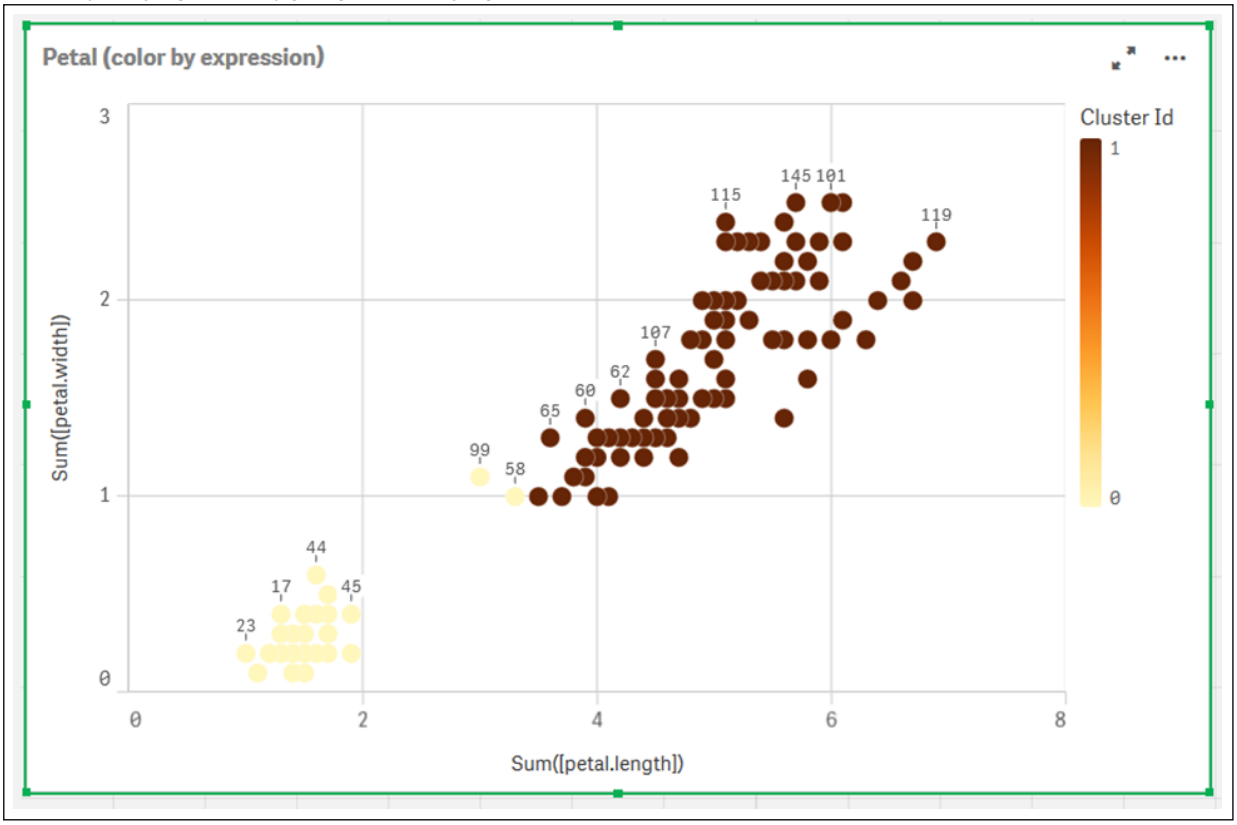
5. Grafik için **Görünüş**'ü yapılandırma:
    i. **Renkler ve gösterge** bölümünde, **Renkler** için **Özel**'i seçin.
    ii. Grafiği **İfadeye göre** renklendirmek için seçin.
    iii. **İfade** için şunu girin: *kmeansnd($(KmeansPetalClusters),$(KmeansNumberIterations), Sum ([petal.length]), Sum([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    KmeansPetalClusters öğesinin, *2* olarak ayarladığımız değişken olduğunu unutmayın.
    KmeansNumberIterations, *1* olarak ayarladığımız değişkendir.
    Alternatif olarak şunu girin: *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
    iv. **İfade bir renk kodudur** onay kutusunun seçimini kaldırın.
    v. **Etiket** için şunu girin: *Küme Kimliği*

Yaprak (ifadeye göre renk) grafiğinin görünüş ayarları


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
fx
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

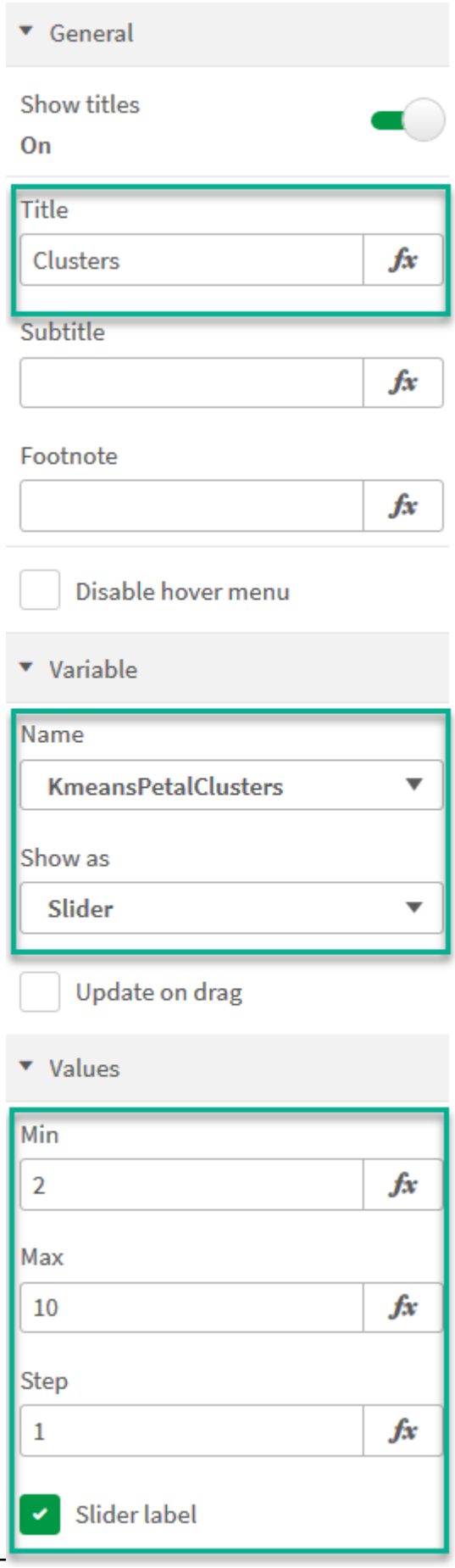
Grafikteki iki küme, KMeans ifadesine göre renklendirilir.

Yaprak (ifadeye göre renk) grafiğinde ifadeye göre renklendirilen kümeler



6. Küme sayısı için bir **Değişken girişi** kutusu ekleyin.
    i. **Varlıklar** panelindeki **Özel nesneler** bölümünde **Qlik Gösterge Paneli paketi**'ni seçin. Gösterge paneli paketine erişimimiz olmasaydı da oluşturduğumuz değişkeni kullanarak veya ifadede tamsayı olarak doğrudan küme sayısını değiştirebilirdik.
    ii. Bir **Değişken girişi** kutusunu sayfaya sürükleyin.
    iii. **Görünüm** bölümünde **Genel**'e tıklayın.
    iv. **Başlık** için şunu girin: *Kümeler*
    v. **Değişken**'e tıklayın.
    vi. **Ad** için şu değişkeni seçin: *KmeansPetalClusters*.
    vii. **Farklı göster** için **Kaydırıcı**'yı seçin.
    viii. **Değerler**'i seçin ve ayarları gerektiği şekilde yapılandırın.

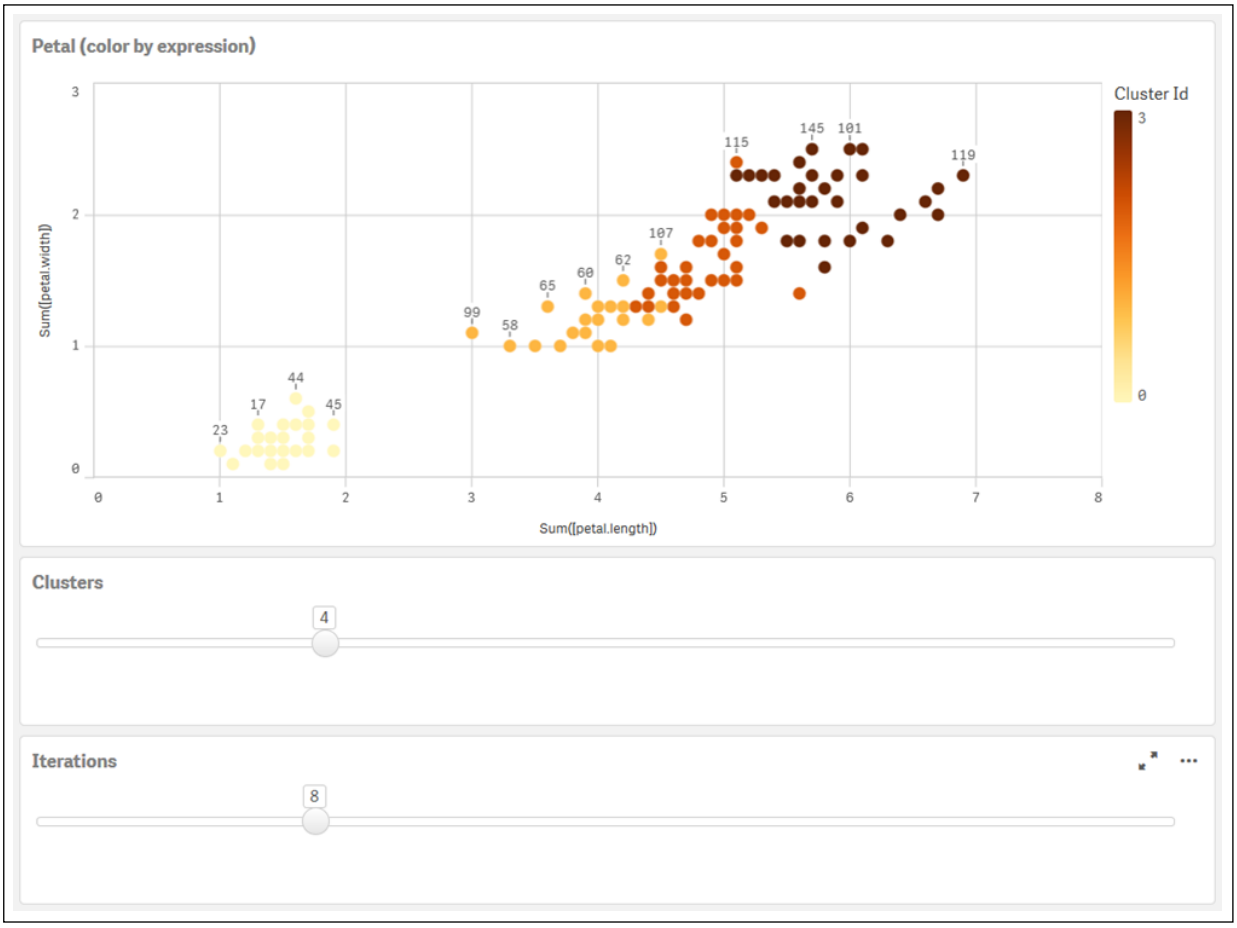
Kümeler değişkeni giriş kutusu görünümü


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. Yineleme sayısı için bir **Değişken girişi** kutusu ekleyin.
    i. Bir **Değişken girişi** kutusunu sayfaya sürükleyin.
    ii. **Görünüm** bölümünde **Genel**'i seçin.
    iii. **Başlık** için şunu girin: *Yinelemeler*
    iv. **Görünüm** bölümünde **Değişken**'i seçin.
    v. **Ad** bölümünde şu değişkeni seçin: *KmeansNumberIterations*.
    vi. Gerektiği şekilde ek ayarları yapılandırın.

Artık değişken giriş kutularındaki kaydırıcıları kullanarak küme ve yineleme sayısını değiştirebiliriz.

Yaprak (ifadeye göre renk) grafiğinde ifadeye göre renklendirilen kümeler

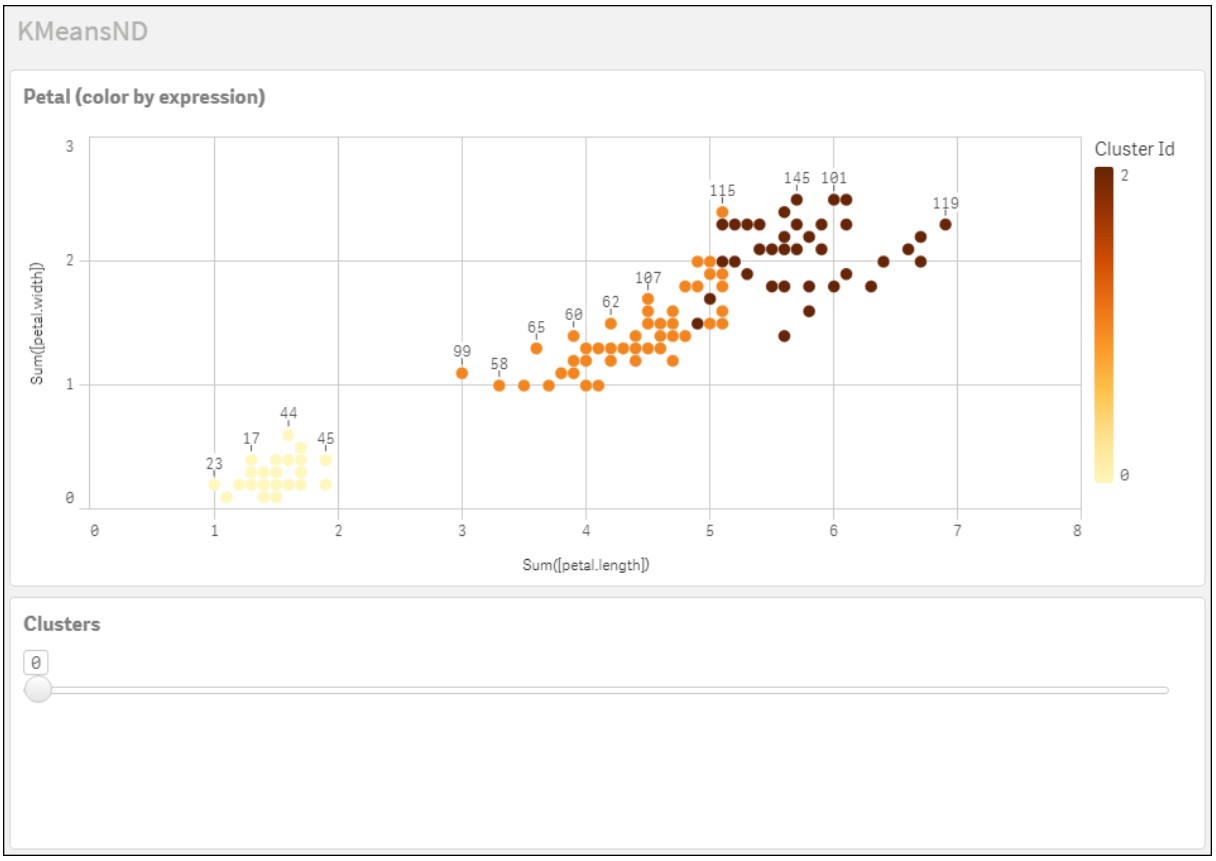


Otomatik kümeleme

KMeans işlevleri, derinlik farkı (DeD) adlı bir yöntem kullanılarak otomatik kümelemeyi destekler. Bir kullanıcı, küme sayısı için 0 değerini ayarladığında, o veri kümesi için optimum küme sayısı belirlenir. Küme sayısı (*k*) için bir tamsayı, belirtik şekilde döndürülmese de KMeans algoritması içinde hesaplanır. Örneğin, *KmeansPetalClusters* değeri için işlevde 0 değeri belirtilirse veya bir değişken giriş kutusu aracılığıyla

ayarlanırsa optimum bir küme sayısına dayalı olarak veri kümesi için küme atamaları otomatik şekilde hesaplanır. Iris veri kümesi söz konusu olduğunda, küme sayısı için 0 değeri seçilirse algoritma, bu veri kümesi için optimum bir küme sayısını (3) belirler (otomatik kümeler).

K-Ortalamaları derinlik farkı yöntemi, (k) değeri 0 olarak ayarlandığı andaki optimum küme sayısını belirler.



Iris veri kümesi: Qlik Sense içinde veri yükleme düzenleyicisi için satır içi yükleme

```
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 - grafik fonksiyonu

KMeansCentroid2D(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin istenen koordinatını görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar sırasıyla coordinate_1 ve coordinate_2 parametreleri tarafından belirlenir. Bunların her ikisi de toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir. Veriler isteğe bağlı olarak norm parametresi ile normalleştirilebilir.

KMeansCentroid2D, veri noktası başına tek bir değer döndürür. Döndürülen değer, ikili değerdir ve veri noktasının atanmış olduğu küme merkezine karşılık gelen konumun koordinatlarından biridir.

Söz Dizimi:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [,
norm])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
num_clusters	Küme sayısını belirten tamsayı.
coordinate_no	Sendroidlerin istenen koordinat numarası (örneğin, x, y veya z eksenine karşılık gelen).
coordinate_1	Birinci koordinatı (genellikle grafikten oluşturulabilen dağılım grafiğinin x ekseni) hesaplayan toplama. coordinate_2 adlı ek parametre, ikinci koordinatı hesaplar.
norm	KMeans kümelemesinden önce veri kümelerine uygulanan isteğe bağlı normalleştirme yöntemi. Olası değerler: Normalleştirme yok için 0 veya "none" z puanı normalleştirme için "zscore" Min-maks normalleştirme için 2 veya "minmax" Hiç parametre sağlanmadıysa veya sağlanan parametre yanlışsa, normalleştirme yok uygulanır. Z puanı, özellik ortalamasına ve standart sapmaya göre verileri normalleştirir. Z puanı, her özelliğin aynı ölçeğe sahip olmasını sağlamaz, ancak aykırı değerlerle çalışırken min-maks'tan daha iyi bir yaklaşımdır. Min-maks normalleştirme, her birinin minimum ve maksimum değerlerini alarak ve her bir veri noktasını yeniden hesaplayarak özelliklerin aynı ölçeğe sahip olmasını sağlar.

Otomatik kümeleme

KMeans işlevleri, derinlik farkı (DeD) adlı bir yöntem kullanılarak otomatik kümelemeyi destekler. Bir kullanıcı, küme sayısı için 0 değerini ayarladığında, o veri kümesi için optimum küme sayısı belirlenir. Küme sayısı (*k*) için bir tamsayı, belirtik şekilde döndürülmese de KMeans algoritması içinde hesaplanır. Örneğin, *KmeansPetalClusters* değeri için işlevde 0 değeri belirtilirse veya bir değişken giriş kutusu aracılığıyla ayarlanırsa optimum bir küme sayısına dayalı olarak veri kümesi için küme atamaları otomatik şekilde hesaplanır.

KMeansCentroidND - grafik fonksiyonu

KMeansCentroidND(), k-ortalama kümelemesi uygulayarak grafiğin satırlarını değerlendirir ve her bir grafik satırı için bu veri noktasının atandığı kümenin istenen koordinatını görüntüler. Kümeleme algoritması tarafından kullanılan sütunlar, n sütuna kadar sırasıyla coordinate_1, coordinate_2 vb. parametreleri tarafından belirlenir. Bunların tümü toplamadır. Oluşturulan küme sayısı, num_clusters parametresi tarafından belirlenir.

KMeansCentroidND, satır başına tek bir değer döndürür. Döndürülen değer, ikili değerdir ve veri noktasının atanmış olduğu küme merkezine karşılık gelen konumun koordinatlarından biridir.

Söz Dizimi:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Dönüş verileri türü: dual

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
num_clusters	Küme sayısını belirten tamsayı.
num_iter	Yeniden başlatılmış küme merkezleri ile kümeleme yinelemesi sayısı.
coordinate_no	Sendroidlerin istenen koordinat numarası (örneğin, x, y veya z eksenine karşılık gelen).
coordinate_1	Birinci koordinatı (genellikle grafikten oluşturulabilen dağılım grafiğinin x ekseni) hesaplayan toplama. Ek parametreler ikinci, üçüncü ve dördüncü koordinatları vb. hesaplar.

Otomatik kümeleme

KMeans işlevleri, derinlik farkı (DeD) adlı bir yöntem kullanılarak otomatik kümelemeyi destekler. Bir kullanıcı, küme sayısı için 0 değerini ayarladığında, o veri kümesi için optimum küme sayısı belirlenir. Küme sayısı (*k*) için bir tamsayı, belirtik şekilde döndürülmese de KMeans algoritması içinde hesaplanır. Örneğin, *KmeansPetalClusters* değeri için işlevde 0 değeri belirtilirse veya bir değişken giriş kutusu aracılığıyla ayarlanırsa optimum bir küme sayısına dayalı olarak veri kümesi için küme atamaları otomatik şekilde hesaplanır.

5.23 İstatistiksel dağıtım fonksiyonları

İstatistiksel dağılım DIST fonksiyonları, sağlanan değer ile verilen dağılım noktasında dağılım fonksiyonunun olasılığını hesaplar. INV fonksiyonları değeri hesaplar (dağılımın olasılığı verildiğinde). Buna karşın, istatistiksel toplama işlevlerinin grupları, çeşitli istatistiksel hipotez testleri için istatistiksel test değerleri serisinin toplanmış değerlerini hesaplar.

Aşağıda açıklanan istatistiksel dağılım fonksiyonlarının tümü, Cephes fonksiyon kitaplığı kullanılarak Qlik Sense içinde uygulanır. Referanslar ve kullanılan algoritmalar, doğruluk vs. hakkında ayrıntılar için bkz. Cephes library. Cephes fonksiyon kütüphanesi izinle kullanılır.

Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

İstatistiksel dağılım fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

CHIDIST

CHIDIST(), chi^2 dağılımının tek kuyruklu olasılığını döndürür. chi^2 dağılımı bir ch^{i2} testi ile ilişkilidir.

```
CHIDIST (value, degrees_freedom)
```

CHIINV

CHIINV(), chi^2 dağılımının tek kuyruklu olasılığının tersini döndürür.

```
CHIINV (prob, degrees_freedom)
```

NORMDIST

NORMDIST(), belirtilen ortalama ve standart sapma için kümülatif normal dağılımı döndürür. mean = 0 ve standard_dev = 1 ise fonksiyon standart normal dağılımı döndürür.

```
NORMDIST (value, mean, standard_dev)
```

NORMINV

NORMINV(), belirtilen ortalama ve standart sapma için normal kümülatif dağılımın tersini döndürür.

```
NORMINV (prob, mean, standard_dev)
```

TDIST

TDIST(), bir sayısal değerin olasılığı hesaplanacak olan t öğesinin hesaplanmış değeri olduğu Öğrencinin t dağılımı için olasılığı döndürür.

```
TDIST (value, degrees_freedom, tails)
```

TINV

TINV(), Öğrencinin t dağılımının t değerini, olasılığın ve serbestlik derecelerinin bir fonksiyonu olarak döndürür.

```
TINV (prob, degrees_freedom)
```

FDIST
FDIST(), F olasılık dağılımını döndürür.

```
FDIST (value, degrees_freedom1, degrees_freedom2)
```

FINV
FINV(), F olasılık dağılımının tersini döndürür.

```
FINV (prob, degrees_freedom1, degrees_freedom2)
```

Ayrıca bkz.

İstatistiksel toplama işlevleri (page 262)

CHIDIST

CHIDIST(), chi^2 dağılımının tek kuyruklu olasılığını döndürür. chi^2 dağılımı bir ch^{i2} testi ile ilişkilidir.

Söz Dizimi:

```
CHIDIST(value, degrees_freedom)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Dağılımı değerlendirmek istediğiniz değer. Değer negatif olmamalıdır.
degrees_freedom	Serbestlik derecesinin sayısını belirten bir pozitif tamsayı.

Bu fonksiyon **CHIINV** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
CHIDIST(8, 15)	0,9238 döndürür

CHIINV

CHIINV(), chi^2 dağılımının tek kuyruklu olasılığının tersini döndürür.

Söz Dizimi:

```
CHIINV(prob, degrees_freedom)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
prob	chi^2 dağılımı ile ilişkili bir olasılık. Bu, 0 ile 1 arasında bir sayı olmalıdır.
degrees_freedom	Serbestlik derecesinin sayısını belirten bir tamsayı.

Bu fonksiyon **CHIDIST** fonksiyonuyla aşağıdaki şekilde ilişkilidir:

```
If prob = CHIDIST(value,df), then CHIINV(prob, df) = value
```

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
CHIINV(0.9237827, 15)	8,0000 döndürür

FDIST

FDIST(), F olasılık dağılımını döndürür.

Söz Dizimi:

```
FDIST(value, degrees_freedom1, degrees_freedom2)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Dağılımı değerlendirmek istediğiniz değer. **Value** negatif olmamalıdır.
degrees_freedom1	Pay serbestlik derecesinin sayısını belirten bir pozitif tamsayı.
degrees_freedom2	Payda serbestlik derecesinin sayısını belirten bir pozitif tamsayı.

Bu fonksiyon **FINV** fonksiyonuyla aşağıdaki şekilde ilişkilidir:

```
If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value
```

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
FDIST(15, 8, 6)	0,0019 döndürür

FINV

FINV(), F olasılık dağılımının tersini döndürür.

Söz Dizimi:

```
FINV(prob, degrees_freedom1, degrees_freedom2)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
prob	F olasılık dağılımıyla ilişkili bir olasılıktır ve 0 ile 1 arasında bir sayı olmalıdır.
degrees_freedom	Serbestlik derecesinin sayısını belirten bir tamsayı.

Bu fonksiyon **FDIST** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
FINV(0.0019369, 8, 6)	15,0000 döndürür

NORMDIST

NORMDIST(), belirtilen ortalama ve standart sapma için kümülatif normal dağılımı döndürür. mean = 0 ve standard_dev = 1 ise fonksiyon standart normal dağılımı döndürür.

Söz Dizimi:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Dağılımı değerlendirmek istediğiniz değer.
mean	Dağılımın aritmetik ortalamasını belirten isteğe bağlı değer. Bu bağımsız değişkeni belirtmezseniz varsayılan değer 0 olur.
standard_dev	Dağılımın standart sapmasını belirten isteğe bağlı pozitif değer. Bu bağımsız değişkeni belirtmezseniz varsayılan değer 1 olur.
cumulative	İsteğe bağlı olarak standart bir normal dağılımı veya kümülatif dağılımı kullanmayı seçebilirsiniz. 0 = standart normal dağılım 1 = kümülatif dağılım (varsayılan)

Bu fonksiyon **NORMINV** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
NORMDIST(0.5, 0, 1)	0,6915 döndürür

NORMINV

NORMINV(), belirtilen ortalama ve standart sapma için normal kümülatif dağılımın tersini döndürür.

Söz Dizimi:

```
NORMINV(prob, mean, standard_dev)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
prob	Normal dağılım ile ilişkili bir olasılık. Bu, 0 ile 1 arasında bir sayı olmalıdır.
mean	Dağılımın aritmetik ortalamasını belirten bir değer.
standard_dev	Dağılımın standart sapmasını belirten bir pozitif değer.

Bu fonksiyon **NORMDIST** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
NORMINV(0.6914625, 0, 1)	0,5000 döndürür

TDIST

TDIST(), bir sayısal değerin olasılığı hesaplanacak olan t öğesinin hesaplanmış değeri olduğu Öğrencinin t dağılımı için olasılığı döndürür.

Söz Dizimi:

```
TDIST(value, degrees_freedom, tails)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
value	Dağılımı değerlendirmek istediğiniz değer (negatif olmamalıdır).
degrees_freedom	Serbestlik derecesinin sayısını belirten bir pozitif tamsayı.
tails	1 (tek kuyruklu dağılım) veya 2 (iki kuyruklu dağılım) olmalıdır.

Bu fonksiyon **TINV** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Örnekler ve sonuçlar:

Örnek	Sonuç
TDIST(1, 30, 2)	0,3253 döndürür

TINV

TINV(), Öğrencinin t dağılımının t değerini, olasılığın ve serbestlik derecelerinin bir fonksiyonu olarak döndürür.

Söz Dizimi:

```
TINV(prob, degrees_freedom)
```

Dönüş verileri türü: sayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
prob	t dağılımıyla ilişkili iki kuyruklu bir olasılık. Bu, 0 ile 1 arasında bir sayı olmalıdır.
degrees_freedom	Serbestlik derecesinin sayısını belirten bir tamsayı.

Sınırlamalar:

Tüm bağımsız değişkenler sayısal olmalıdır, aksi takdirde NULL döndürülür.

Bu fonksiyon **TDIST** fonksiyonuyla aşağıdaki şekilde ilişkilidir:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Örnekler ve sonuçlar:

Örnek	Sonuç
TINV(0.3253086, 30)	1,0000 döndürür

5.24 Dize fonksiyonları

Bu bölümde, dizeleri işlemeye ve yönlendirmeye yönelik fonksiyonlar açıklanmaktadır.

Yalnızca veri kod dosyasında kullanılabilen **Evaluate** fonksiyonu dışında tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

Dize fonksiyonlarına genel bakış

Genel bakıştan sonra her bir fonksiyon daha ayrıntılı olarak açıklanmaktadır. Ayrıca, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Capitalize

Capitalize(), dizeyi tüm sözcüklerin baş harfi büyük olacak şekilde döndürür.

```
Capitalize (text)
```

Chr

Chr(), girdi tamsayısına karşılık gelen Unicode karakterini döndürür.

```
Chr (int)
```

Evaluate

Evaluate(), giriş metninin geçerli bir Qlik Sense ifadesi olarak değerlendirilip değerlendirilemeyeceğini bulur ve öyleyse, bu ifadenin değerini bir dize olarak döndürür. Giriş dizesi geçerli bir ifade değilse NULL döndürülür.

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf(), sağlanan bir karakter kümesinden herhangi bir karakterin oluş konumunu bulmak için bir dize arar. Üçüncü bir bağımsız değişken (1'den büyük değerli) sağlanmadığı takdirde, arama kümesinden herhangi bir karakterin ilk oluş konumu döndürülür. Herhangi bir eşleşme bulunamazsa **0** sonucu döndürülür.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128(), birleştirilmiş giriş ifade değerlerinin 128 bitlik karmasını verir. Sonuç, 22 karakterli bir dizedir.

```
Hash128 (expr{, expression})
```

Hash160

Hash160(), birleştirilmiş girdi ifade değerlerinin 160 bitlik karmasını döndürür. Sonuç, 27 karakterli bir dizedir.

```
Hash160 (expr{, expression})
```

Hash256

Hash256(), birleştirilmiş girdi ifade değerlerinin 256 bitlik karmasını döndürür. Sonuç, 43 karakterli bir dizedir.

```
Hash256 (expr{, expression})
```

Index

Index(), sağlanan bir alt dizenin n. oluşumunun başlangıç konumunu bulmak için bir dizeyi arar. İsteğe bağlı üçüncü bir bağımsız değişken n değerini sağlar; atlanması halinde bu 1 olur. Negatif bir değer dizenin sonundan itibaren arar. Dizedeki konumlar **1**'den başlayarak ve artarak numaralandırılır.

```
Index  (text, substring[, count])
```

KeepChar

KeepChar(), ikinci dize "keep_chars" içinde yer ALMAYAN karakterler hariç olmak üzere ilk dize 'text'ten oluşan bir dize döndürür.

```
KeepChar (text, keep_chars)
```

Left

Left(), karakter sayısının ikinci bağımsız değişken tarafından belirlendiği ve girdi dizesinin ilk (en soldaki) karakterlerinden oluşan bir dize döndürür.

```
Left (text, count)
```

Len

Len(), girdi dizesinin uzunluğunu döndürür.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() iki dize arasında Levenshtein mesafesi döndürür. Bu, bir dizeyi diğeriyle değiştirmek için gereken minimum tek karakterli düzenleme (ekleme, silme veya değiştirme) sayısı olarak tanımlanır. Fonksiyon, fuzzy dize karşılaştırmaları için kullanışlıdır.

```
LevenshteinDist (text1, text2)
```

Lower

Lower(), girdi dizesindeki tüm karakterleri küçük harfe dönüştürür.

```
Lower (text)
```

LTrim

LTrim(), girdi dizesini öndeki boşlukları kırpılmış olarak döndürür.

```
LTrim (text)
```

Mid

Mid(), ikinci bağımsız değişken 'start' tarafından tanımlanan karakterin konumundan başlayarak ve üçüncü bağımsız değişken 'count' tarafından tanımlanan karakterlerin sayısını döndürerek giriş dizesinin bir bölümünü döndürür. 'count' atlanırsa, dizenin geri kalanı döndürülür. Giriş dizesindeki ilk karakter 1 olarak numaralandırılır.

```
Mid (text, start[, count])
```

Ord

Ord(), girdi dizesinin ilk karakterinin Unicode kod noktası numarasını döndürür.

```
Ord (text)
```

PurgeChar

PurgeChar(), girdi dizesinde ('text') yer alan karakterlerden oluşan ve ikinci bağımsız değişkende ('remove_chars') görülen karakterlerin hariç tutulduğu bir dize döndürür.

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat(), girdi dizesinin ikinci bağımsız değişkenin tanımladığı tekrar sayısı kadar yinelenmesinden oluşan bir dize oluşturur.

```
Repeat (text[, repeat_count])
```

Replace

Replace(), giriş dizesi içindeki verilen bir alt dizenin tüm oluşlarını başka bir alt dizeyle değiştirildikten sonra oluşan dizeyi döndürür. Bu fonksiyon özyinelemesizdir ve soldan sağa doğru çalışır.

```
Replace (text, from_str, to_str)
```

Right

Right(), karakter sayısının ikinci bağımsız değişken tarafından belirlendiği ve giriş dizesinin son (en sağdaki) karakterlerinden oluşan bir dize döndürür.

```
Right (text, count)
```

RTrim

RTrim(), girdi dizesini sondaki boşlukları kırpılmış olarak döndürür.

```
RTrim (text)
```

SubField

SubField(), orijinal kayıt alanlarının bir ayırıcıyla ayrılmış iki veya daha fazla bölümden oluştuğu bir üst dize alanından alt dize bileşenlerini ayıklamak için kullanılır.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount(), girdi dizesi metninde belirtilen alt dizenin oluşum sayısını döndürür. Eşleşme yoksa, 0 sonucu döndürülür.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween(), girdi dizesinde ayırıcılar olarak belirtilen karakterler arasında olan metni döndürür.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim(), girdi dizesini öndeki ve sondaki boşlukları kırpılmış olarak döndürür.

```
Trim (text)
```

Upper

Upper(), ifadedeki tüm metin karakterleri için giriş dizesindeki tüm karakterleri büyük harfe dönüştürür. Sayılar ve semboller yok sayılır.

```
Upper (text)
```

Capitalize

Capitalize(), dizeyi tüm sözcüklerin baş harfi büyük olacak şekilde döndürür.

Söz Dizimi:

```
Capitalize(text)
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
`Capitalize ( 'star trek' )`	'Star Trek' döndürür
`Capitalize ( 'AA bb cC Dd')`	'Aa Bb Cc Dd' döndürür

Örnek: Kod dosya

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Sonuç

Dize	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr(), girdi tamsayısına karşılık gelen Unicode karakterini döndürür.

Söz Dizimi:

```
Chr(int)
```

Dönüş verileri türü: dize

Örnekler ve sonuçlar:

Örnek	Sonuç
Chr(65)	'A' dizesini döndürür

Örnek	Sonuç
Chr(163)	'£' dizesini döndürür
Chr(35)	'#' dizesini döndürür

Evaluate

Evaluate(), giriş metninin geçerli bir Qlik Sense ifadesi olarak değerlendirilip değerlendirilemeyeceğini bulur ve öyleyse, bu ifadenin değerini bir dize olarak döndürür. Giriş dizesi geçerli bir ifade değilse NULL döndürülür.

Söz Dizimi:

```
Evaluate(expression_text)
```

Dönüş verileri türü: dual

Bu dize fonksiyonu grafik ifadelerinde kullanılamaz.

Örnekler ve sonuçlar:

Fonksiyon örneği	Sonuç
`Evaluate ( 5 * 8 )`	'40' döndürür

Komut dosyası örneği

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Sonuç

Dize	Değerlendirildi
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf(), sağlanan bir karakter kümesinden herhangi bir karakterin oluş konumunu bulmak için bir dize arar. Üçüncü bir bağımsız değişken (1'den büyük değerli) sağlanmadığı takdirde, arama kümesinden herhangi bir karakterin ilk oluş konumu döndürülür. Herhangi bir eşleşme bulunamazsa **0** sonucu döndürülür.

Söz Dizimi:

```
FindOneOf(text, char_set[, count])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
char_set	text içinde aranacak bir dizi karakter.
count	Karakterlerden herhangi birinin hangi oluşunun aranacağını tanımlar. Örneğin, 2 değeri ikinci oluşu arar.

Örnek: Grafik ifadeleri

Örnek	Sonuç
`FindOneOf( 'my example text string', 'et%s')`	'e' örnek dizedeki dördüncü karakter olduğu için '4' döndürür.
`FindOneOf( 'my example text string', 'et%s', 3)`	Arama e, t, % veyas karakterlerinden herhangi biri için yapıldığından '12' döndürür ve"t" örnek dizenin 12. konumundaki üçüncü oluşumdur.
`FindOneOf( 'my example text string', '¤%&')`	¤, % veya & karakterlerinin hiçbiri örnek dizede olmadığı için '0' döndürür.

Örnek: Kod dosya

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Sonuç

SearchFor	Occurrence	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128(), birleştirilmiş giriş ifade değerlerinin 128 bitlik karmasını verir. Sonuç, 22 karakterli bir dizedir.

Söz Dizimi:

```
Hash128(expr{, expression})
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
Hash128 ('abc', 'xyz', '123')	'MA&5]6+3=:>:>G%S<U*S2+' döndürür.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'A+' döndürür.

Örnek: Kod dosya

```
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 ];
```

Sonuç

Bölge	Yıl	Ay	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(), birleştirilmiş girdi ifade değerlerinin 160 bitlik karmasını döndürür. Sonuç, 27 karakterli bir dizedir.

Söz Dizimi:

```
Hash160(expr{, expression})
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
Hash160 ('abc', 'xyz', '123')	'MA&5]6+3=:>;>G%S<U*S2I:`=X*' döndürür.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$' döndürür.

Örnek: Kod dosya

```
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 ];
```

Sonuç

Bölge	Yıl	Ay	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(), birleştirilmiş girdi ifade değerlerinin 256 bitlik karmasını döndürür. Sonuç, 43 karakterli bir dizedir.

Söz Dizimi:

```
Hash256(expr{, expression})
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
Hash256 ('abc', 'xyz', '123')	'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ' döndürür.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB[:0ZP=+Z`F:' döndürür.

Örnek: Kod dosya

```
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 ];
```

Sonuç

Bölge	Yıl	Ay	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(), sağlanan bir alt dizenin n. oluşumunun başlangıç konumunu bulmak için bir dizeyi arar. İsteğe bağlı üçüncü bir bağımsız değişken n değerini sağlar; atlanması halinde bu 1 olur. Negatif bir değer dizenin sonundan itibaren arar. Dizedeki konumlar **1**'den başlayarak ve artarak numaralandırılır.

Söz Dizimi:

```
Index(text, substring[, count])
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
substring	text içinde aranacak bir karakter dizesi.
count	**substring** öğesinin hangi oluşunun aranacağını tanımlar. Örneğin, 2 değeri ikinci oluşu arar.

Örnekler ve sonuçlar:

Örnek	Sonuç
Index('abcdefg', 'cd')	3 döndürür
Index('abcdabcd', 'b', 2)	6 döndürür ('b'nin ikinci oluşu)
Index('abcdabcd', 'b',-2)	2 döndürür (sondan başlayarak 'b'nin ikinci oluşu)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	1997 döndürür
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	07 döndürür

Örnek: Kod

```
T1: Load  *,  index(String, 'cd') as Index_CD,              // returns 3  in Index_CD index
(String, 'b') as Index_B,               // returns 2 in Index_B index(String, 'b', -1) as
Index_B2;           // returns 2 or 6 in Index_B2 Load * inline [ String abcdefg abcdabcd ];
```

KeepChar

KeepChar(), ikinci dize "keep_chars" içinde yer ALMAYAN karakterler hariç olmak üzere ilk dize 'text'ten oluşan bir dize döndürür.

Söz Dizimi:

```
KeepChar(text, keep_chars)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
keep_chars	text içindeki tutulacak karakterleri içeren dize.

Örnek: Grafik ifadeleri

Örnek	Sonuç
`KeepChar ( 'a1b2c3','123' )`	'123' döndürür.
`KeepChar ( 'a1b2c3','1234' )`	'123' döndürür.
`KeepChar ( 'a1b22c3','1234' )`	'1223' döndürür.
`KeepChar ( 'a1b2c3','312' )`	'123' döndürür.

Örnek: Komut dosyası

```
T1: Load  *,  keepchar(String1, String2) as KeepChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Sonuçlar

Yükleme kodundaki *KeepChar* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

String1	String2	KeepChar
a1b2c3	123	123

Ayrıca bkz.

PurgeChar (page 803)

Left

Left(), karakter sayısının ikinci bağımsız değişken tarafından belirlendiği ve girdi dizesinin ilk (en soldaki) karakterlerinden oluşan bir dize döndürür.

Söz Dizimi:

```
Left(text, count)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
text	Orijinal dize.
count	**text** dizesinin sol bölümünden dahil edilecek karakter sayısını tanımlar.

Örnek: Grafik ifadesi

Örnek	Sonuç
`Left('abcdef', 3)`	'abc' sonucunu döndürür

Örnek: Kod dosya

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Sonuç

Yükleme kodundaki *Left* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

Metin	Başlat	Sol
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Ayrıca bkz., daha karmaşık dize analizine olanak sağlayan *Index (page 796)*.

Len

Len(), girdi dizesinin uzunluğunu döndürür.

Söz Dizimi:

```
Len(text)
```

Dönüş verileri türü: tamsayı

Örnek: Grafik ifadesi

Örnek	Sonuç
`Len('Peter')`	'5' döndürür

Örnek: Kod dosya

```
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 ];
```

Sonuç

Dize	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() iki dize arasında Levenshtein mesafesi döndürür. Bu, bir dizeyi diğeriyle değiştirmek için gereken minimum tek karakterli düzenleme (ekleme, silme veya değiştirme) sayısı olarak tanımlanır. Fonksiyon, fuzzy dize karşılaştırmaları için kullanışlıdır.

Söz Dizimi:

```
LevenshteinDist(text1, text2)
```

Dönüş verileri türü: tamsayı

Örnek: Grafik ifadesi

Örnek	Sonuç
`LevenshteinDist('Kitten','Sitting')`	'3' döndürür

Örnek: Yükleme kodu

Komut dosyası

```
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 ᵾᵾᵾ ];  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;
```

Sonuç

Kimlik	Dize_1	Dize_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Bold	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(), girdi dizesindeki tüm karakterleri küçük harfe dönüştürür.

Söz Dizimi:

```
Lower(text)
```

Dönüş verileri türü: dize

Örnek: Grafik ifadesi

Örnek	Sonuç
`Lower('abcD')`	'abcd' döndürür

Örnek: Kod dosya

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Sonuç

Dize	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim(), girdi dizesini öndeki boşlukları kırpılmış olarak döndürür.

Söz Dizimi:

```
LTrim(text)
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
`LTrim( ' abc' )`	'abc' döndürür
`LTrim( 'abc ' )`	'abc ' döndürür

Örnek: Komut dosyası

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

"Set verbatim=1" deyimi, ltrim fonksiyonunun gösterilmesinden önce boşlukların otomatik olarak kırpılmamasını sağlamak için örneğe dahil edilmiştir. Daha fazla bilgi için bkz. Verbatim (page 147).

Sonuç

Dize	StringLength	LtrimStringLength
def	6	5
abc	10	7

Ayrıca bkz.

- *RTrim (page 806)*

Mid

Mid(), ikinci bağımsız değişken 'start' tarafından tanımlanan karakterin konumundan başlayarak ve üçüncü bağımsız değişken 'count' tarafından tanımlanan karakterlerin sayısını döndürerek giriş dizesinin bir bölümünü döndürür. 'count' atlanırsa, dizenin geri kalanı döndürülür. Giriş dizesindeki ilk karakter 1 olarak numaralandırılır.

Söz Dizimi:

```
Mid(text, start[, count])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
start	text içinde dahil edilecek ilk karakterin konumunu tanımlayan tamsayı.
count	Çıkış dizesinin dize uzunluğunu tanımlar. Atlandığı takdirde, **start** ile tanımlanan konumdan itibaren tüm karakterler dahil edilir.

Örnek: Grafik ifadeleri

Örnek	Sonuç
`Mid('abcdef',3 )`	'cdef' döndürür
`Mid('abcdef',3, 2 )`	'cd' döndürür

Örnek: Kod dosya

```
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 ];
```

Sonuç

Yükleme kodundaki *Mid* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

Metin	Başlat	Mid1	Sayım	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Ayrıca bkz.

Index (page 796)

Ord

Ord(), girdi dizesinin ilk karakterinin Unicode kod noktası numarasını döndürür.

Söz Dizimi:

```
Ord(text)
```

Dönüş verileri türü: tamsayı

Örnekler ve sonuçlar:

Örnek: Grafik ifadesi

Örnek	Sonuç
`Ord('A')`	65 tamsayısını döndürür.
`Ord('Ab')`	65 tamsayısını döndürür.

Örnek: Komut dosyası

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Sonuç:

Çince	Western	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar(), girdi dizesinde ('text') yer alan karakterlerden oluşan ve ikinci bağımsız değişkende ('remove_chars') görülen karakterlerin hariç tutulduğu bir dize döndürür.

Söz Dizimi:

```
PurgeChar(text, remove_chars)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
remove_chars	text içindeki çıkarılacak karakterleri içeren dize.

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
`PurgeChar ( 'a1b2c3','123' )`	"abc" sonucunu döndürür.
`PurgeChar ( 'a1b2c3','312' )`	"abc" sonucunu döndürür.

Örnek: Komut dosyası

```
T1: Load  *,  purgechar(String1, String2) as PurgeChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Sonuçlar

Yükleme kodundaki *PurgeChar* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

String1	String2	PurgeChar
a1b2c3	123	abc

Ayrıca bkz.

- *KeepChar (page 797)*

Repeat

Repeat(), girdi dizesinin ikinci bağımsız değişkenin tanımladığı tekrar sayısı kadar yinelenmesinden oluşan bir dize oluşturur.

Söz Dizimi:

```
Repeat(text[, repeat_count])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
repeat_count	**text** dizesindeki karakterlerin çıkış dizesinde kaç kez yineleneceğini tanımlar.

Örnek: Grafik ifadesi

Örnek	Sonuç
`Repeat( ' * ', rating )` when **rating** = 4	'****' döndürür

Örnek: Kod dosya

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Sonuç

Dize	Yinele
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace(), giriş dizesi içindeki verilen bir alt dizenin tüm oluşlarını başka bir alt dizeyle değiştirildikten sonra oluşan dizeyi döndürür. Bu fonksiyon özyinelemesizdir ve soldan sağa doğru çalışır.

Söz Dizimi:

```
Replace(text, from_str, to_str)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
from_str	Giriş dizesi **text** içinde bir veya daha fazla kez bulunabilen bir dize.
to_str	**text** dizesi içindeki tüm **from_str** oluşlarının yerine geçecek dize.

Örnekler ve sonuçlar:

Örnek	Sonuç
`Replace('abccde','cc','xyz')`	'abxyzde' döndürür

Ayrıca bkz.

Right

Right(), karakter sayısının ikinci bağımsız değişken tarafından belirlendiği ve giriş dizesinin son (en sağdaki) karakterlerinden oluşan bir dize döndürür.

Söz Dizimi:

```
Right(text, count)
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize.
count	**text** dizesinin en sağ kısmından dahil edilecek karakter sayısını tanımlar.

Örnek: Grafik ifadesi

Örnek	Sonuç
`Right('abcdef', 3)`	'def' döndürür

Örnek: Kod dosya

```
T1: Load  *,  right(Text,Start) as Right;        Load * inline [ Text, Start 'abcdef', 3
'2021-07-14', 4 '2021-07-14', 2 ];
```

Sonuç

Yükleme kodundaki *Right* fonksiyonunun kullanılmasından elde edilen çıktıyı gösteren Qlik Sense tablosu.

Metin	Başlat	Sağ
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim(), girdi dizesini sondaki boşlukları kırpılmış olarak döndürür.

Söz Dizimi:

```
RTrim(text)
```

Dönüş verileri türü: dize

Örnek: Grafik ifadeleri

Örnek	Sonuç
`RTrim( ' abc' )`	' abc' döndürür
`RTrim( 'abc ' )`	'abc' döndürür

Örnek: Komut dosyası

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

"Set verbatim=1" deyimi, rtrim fonksiyonunun gösterilmesinden önce boşlukların otomatik olarak kırpılmamasını sağlamak için örneğe dahil edilmiştir. Daha fazla bilgi için bkz. Verbatim (page 147).

Sonuç

Dize	StringLength	RtrimStringLength
def	6	4
abc	10	6

Ayrıca bkz.

- *LTrim (page 801)*

SubField

SubField(), orijinal kayıt alanlarının bir ayırıcıyla ayrılmış iki veya daha fazla bölümden oluştuğu bir üst dize alanından alt dize bileşenlerini ayıklamak için kullanılır.

Subfield() fonksiyonu örneğin, tam adlardan oluşan bir kayıt listesinden adı veya soyadı ayıklamak, bir yol adının bileşen parçalarına ayıklamak veya virgülle ayrılmış tablolardan verileri ayıklamak için kullanılabilir.

Subfield() fonksiyonunu bir **LOAD** deyimi içinde isteğe bağlı field_no parametresini hariç bırakarak kullanırsanız, her bir alt dize için bir tam kayıt üretilir. **Subfield()** kullanılarak birkaç alan yüklenirse, tüm kombinasyonların Kartezyen çarpımları oluşturulur.

Söz Dizimi:

```
SubField(text, delimiter[, field_no ])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
text	Orijinal dize. Bu bir sabit kodlanmış metin, değişken, dolar işareti genişletmesi veya başka bir ifade olabilir.
delimiter	Dizeyi bileşen parçalarına ayıran **text** girişi içindeki bir karakter.
field_no	İsteğe bağlı üçüncü bağımsız değişken, **text** ana dizesinin hangi alt dizelerinin döndürüleceğini belirten bir tamsayıdır. İlk alt dizeyi döndürmek için 1 değerini, ikinci alt dizeyi döndürmek için 2 değerini kullanın ve bu şekilde devam edin. • **field_no** pozitif bir değerse alt dizeler soldan sağa doğru ayıklanır. • **field_no** negatif bir değerse alt dizeler sağdan sola doğru ayıklanır.

Len(), Right(), Left(), Mid() gibi fonksiyonların ve diğer dize fonksiyonlarının karmaşık kombinasyonlarını kullanmak yerine SubField() kullanılabilir.

Örnekler: SubField kullanan kod ve grafik ifadeleri

Örnekler - Kod ve grafik ifadeleri

Temel örnekler

Örnek	Sonuç
`SubField(S, ';' ,2)`	**S** 'abc;cde;efg' ise 'cde' döndürür.
`SubField(S, ';' ,1)`	**S** boş bir dizeyse boş bir dize döndürür.
`SubField(S, ';' ,1)`	**S** ';' ise boş bir dize döndürür.
vMyPath yol adını içeren bir değişkeninizin olduğunu varsayın, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	Metin ve resim grafiğinde şunun gibi bir hesaplama ekleyebilirsiniz: vMyPath değişkeninin sağ tarafındaki üçüncü alt dize olduğu için 'Qlik' ile sonuçlanan `SubField(vMyPath, '\',-3)`.

Kod örneği 1

Komut dosyası

Aşağıdaki kod ifadelerini ve verileri veri yükleme düzenleyicisine yükleyin.

```
FullName:                                    LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames:  Lo
(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table
FullName;
```

Görselleştirme oluşturma

Bir Qlik Sense sayfasında, **Ad**, **Ad** ve **Soyadı** boyutlarıyla bir tablo görselleştirmesi oluşturun.

Sonuç

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Açıklama

SubField() fonksiyonu, **field_no** bağımsız değişkenini 1 olarak ayarlayarak **Ad**'ın ilk alt dizesini ayıklar. **field_no** değeri pozitif olduğundan, alt dizeyi ayıklamak için soldan sağa bir sıra izlenir. İkinci bir fonksiyon çağrısı, **field _no** bağımsız değişkenini -1 olarak ayarlayarak ikinci alt dizeyi ayıklar. Bu, alt dizeyi sağdan sola sıralayarak ayıklar.

Kod örneği 2

Komut dosyası

Aşağıdaki kod ifadelerini ve verileri veri yükleme düzenleyicisine yükleyin.

```
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 '|');
```

Görselleştirme oluşturma

Qlik Sense sayfasında **Araç**, **Oynatıcı** ve **Proje** boyutlarıyla bir tablo görselleştirmesi oluşturun.

Sonuç

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video

Instrument	Player	Project
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Açıklama

Bu örnek, **Subfield()** fonksiyonunu aynı **LOAD** deyimi içerisinden field_no parametresi dışarıda bırakılarak kullanmanın, nasıl tüm kombinasyonların Kartezyen çarpımlarını oluşturduğunu gösterir. Yinelenen kayıt oluşturmaktan kaçınmak için **DISTINCT** seçeneği kullanılır.

SubStringCount

SubStringCount(), girdi dizesi metninde belirtilen alt dizenin oluşum sayısını döndürür. Eşleşme yoksa, 0 sonucu döndürülür.

Söz Dizimi:

```
SubStringCount(text, sub_string)
```

Dönüş verileri türü: tamsayı

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
text	Orijinal dize.
sub_string	**text** giriş dizesi içinde bir kez veya daha çok geçebilen bir dize.

Örnek: Grafik ifadeleri

Örnek	Sonuç
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	'2' döndürür
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	'0' döndürür

Örnek: Kod dosya

```
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 ];
```

Sonuç

Dizeler	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween(), girdi dizesinde ayırıcılar olarak belirtilen karakterler arasında olan metni döndürür.

Söz Dizimi:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Dönüş verileri türü: dize

Bağımsız Değişkenler:

Bağımsız Değişken	Açıklama
text	Orijinal dize.
delimiter1	**text** içinde aranacak ilk sınırlayıcı karakteri (veya dizeyi) belirtir.
delimiter2	**text** içinde aranacak ikinci sınırlayıcı karakteri (veya dizeyi) belirtir.
n	Sınırlayıcı çiftinin hangi oluşu arasında arama yapılacağını tanımlar. Örneğin, 2 değeri sınırlayıcı1 öğesinin ikinci oluşu ile sınırlayıcı2 öğesinin ikinci oluşu arasındaki karakterleri döndürür.

Örnek: Grafik ifadeleri

Örnek	Sonuç
`TextBetween('<abc>', '<', '>')`	'abc' döndürür
`TextBetween('<abc><de>', '<', '>',2)`	'de' döndürür
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Her iki örnek de NULL döndürür. Dizede herhangi bir sınırlayıcı bulunmazsa NULL döndürülür.
`TextBetween('<>', '<', '>')`	Sıfır uzunlukta bir dize döndürür.
`TextBetween('<abc>', '<', '>', 2)`	n sınırlayıcıların kullanılma sayısından daha büyük olduğundan NULL döndürür.

Örnek: Kod dosya

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Sonuç

Metin	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim(), girdi dizesini öndeki ve sondaki boşlukları kırpılmış olarak döndürür.

Söz Dizimi:

```
Trim(text)
```

Dönüş verileri türü: dize

Örnekler ve sonuçlar:

Örnek: Grafik ifadesi

Örnek	Sonuç
`Trim( ' abc' )`	'abc' döndürür
`Trim( 'abc ' )`	'abc' döndürür
`Trim( ' abc ' )`	'abc' döndürür

Örnek: Komut dosyası

```
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');
```

"Set verbatim=1" deyimi, trim fonksiyonunun gösterilmesinden önce boşlukların otomatik olarak kırpılmamasını sağlamak için örneğe dahil edilmiştir. Daha fazla bilgi için bkz. Verbatim (page 147).

Sonuç:

Dize	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper(), ifadedeki tüm metin karakterleri için giriş dizesindeki tüm karakterleri büyük harfe dönüştürür. Sayılar ve semboller yok sayılır.

Söz Dizimi:

```
Upper(text)
```

Dönüş verileri türü: dize

Örnek: Grafik ifadesi

Örnek	Sonuç
`Upper(' abcD')`	'ABCD' döndürür

Örnek: Kod dosya

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Sonuç

Dize	upper(Dize)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Sistem fonksiyonları

Sistem fonksiyonları sistem, cihaz ve Qlik Sense uygulama özelliklerine erişime yönelik fonksiyonlar sağlar.

Sistem fonksiyonlarına genel bakış

Genel bakıştan sonra bazı fonksiyonlar daha ayrıntılı olarak açıklanmaktadır. Bu fonksiyonlar için, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

Author()

Bu fonksiyon geçerli uygulamanın yazar özelliğini içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

> *Yazar özelliği, geçerli Qlik Sense sürümünde ayarlanamaz. QlikView belgesini taşırsanız, yazar özelliği korunacaktır.*

ClientPlatform()

Bu fonksiyon istemci tarayıcısının kullanıcı aracısı dizesini döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

Örnek:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Bu fonksiyon işletim sisteminin döndürdüğü şekliyle bilgisayarın adını içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

Bilgisayarın adı 15 karakterden uzunsa dize yalnızca ilk 15 karakteri içerir.

```
ComputerName( )
```

DocumentName

Bu fonksiyon, geçerli Qlik Sense uygulamasının adını, yolu olmadan ve yalnızca uzantısı olacak şekilde içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
DocumentName( )
```

DocumentPath

Bu fonksiyon, geçerli Qlik Sense uygulamasına giden tam yolu içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
DocumentPath( )
```

Bu fonksiyon, standart modda desteklenmez.

DocumentTitle

Bu fonksiyon, geçerli Qlik Sense uygulamasının başlığını içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
DocumentTitle( )
```

EngineVersion

Bu fonksiyon tam Qlik Sense alt yapı sürümünü bir dize olarak döndürür.

```
EngineVersion ()
```

GetCollationLocale

Bu kod fonksiyonu kullanılan harmanlama yerel ayarının kültür adını döndürür. CollationLocale değişkeni ayarlanmamışsa, asıl kullanıcı makinesi yerel ayarı döndürülür.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField(), boyutun adını döndürür. **Index**, döndürülmesi gereken boyutu belirten isteğe bağlı bir tamsayıdır.

```
GetObjectField - grafik fonksiyonu([index])
```

GetRegistryString

Bu fonksiyon Windows kayıt defterindeki bir anahtarın değerini döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
GetRegistryString(path, key)
```

Bu fonksiyon, standart modda desteklenmez.

IsPartialReload

Bu fonksiyon, geçerli yeniden yükleme kısmiyse -1 (True), değilse 0 (False) değerini döndürür.

```
IsPartialReload ()
```

OSUser

Bu fonksiyon, şu anda bağlı olan kullanıcının adını içeren bir dize döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
OSUser( )
```

Qlik Sense Desktop ve Qlik Sense Mobile Client Managed içinde bu fonksiyon her zaman 'Personal\Me' değerini döndürür.

ProductVersion

Bu fonksiyon tam Qlik Sense sürümü ve derleme numarasını bir dize olarak döndürür.

Bu fonksiyon kullanımdan kaldırılmıştır ve yerine **EngineVersion()** fonksiyonu kullanılmaktadır.

```
ProductVersion ()
```

ReloadTime

Bu fonksiyon son veri yüklemesinin bittiği zaman için bir zaman damgası döndürür. Hem veri kod dosyasında hem de grafik ifadesinde kullanılabilir.

```
ReloadTime( )
```

StateName

StateName(), içinde kullanıldığı görselleştirmenin alternatif durum adını döndürür. StateName, örneğin bir görselleştirmenin durumu değiştirildiğinde bunu yansıtan dinamik metinler ve renkler içeren görselleştirmeler oluşturmak için kullanılabilir. Bu işlev grafik ifadelerinde kullanılabilir, ancak ifadenin başvurduğu durumu belirlemek için kullanılamaz.

```
StateName - grafik fonksiyonu()
```

EngineVersion

Bu fonksiyon tam Qlik Sense alt yapı sürümünü bir dize olarak döndürür.

Söz Dizimi:

```
EngineVersion()
```

IsPartialReload

Bu fonksiyon, geçerli yeniden yükleme kısmiyse - 1 (True), değilse 0 (False) değerini döndürür.

Söz Dizimi:

```
IsPartialReload()
```

ProductVersion

Bu fonksiyon tam Qlik Sense sürümü ve derleme numarasını bir dize olarak döndürür. Bu fonksiyon kullanımdan kaldırılmıştır ve yerine **EngineVersion()** fonksiyonu kullanılmaktadır.

Söz Dizimi:

```
ProductVersion()
```

StateName - grafik fonksiyonu

StateName(), içinde kullanıldığı görselleştirmenin alternatif durum adını döndürür. StateName, örneğin bir görselleştirmenin durumu değiştirildiğinde bunu yansıtan dinamik metinler ve renkler içeren görselleştirmeler oluşturmak için kullanılabilir. Bu işlev grafik ifadelerinde kullanılabilir, ancak ifadenin başvurduğu durumu belirlemek için kullanılamaz.

Söz Dizimi:

```
StateName ()
```

Example 1:

Dinamik Metin

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Dinamik Renkler

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Tablo fonksiyonları

Tablo fonksiyonları, o anda okunan veri tablosuyla ilgili bilgileri döndürür. Tablo adı belirtilmezse ve fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli tablo olduğu varsayılır.

Veri kod dosyasında tüm fonksiyonlar kullanılabilirken, grafik ifadesinde yalnızca **NoOfRows** kullanılabilir.

Tablo fonksiyonlarına genel bakış

Genel bakıştan sonra bazı fonksiyonlar daha ayrıntılı olarak açıklanmaktadır. Bu fonksiyonlar için, söz diziminde fonksiyon adına tıklayarak ilgili fonksiyonun ayrıntılarına hemen erişebilirsiniz.

FieldName

FieldName kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki belirtilen bir sayıya sahip alanın adını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

```
FieldName (field_number ,table_name)
```

FieldNumber

FieldNumber kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki belirtilen bir alanın sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

NoOfFields kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki alanların sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

```
NoOfFields (table_name)
```

NoOfRows

NoOfRows fonksiyonu, daha önce yüklenmiş bir tablo içindeki satırların (kayıtların) sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

```
NoOfRows (table_name)
```

NoOfTables

Bu kod fonksiyonu daha önce yüklenmiş tabloların sayısını döndürür.

```
NoOfTables()
```

TableName

Bu kod fonksiyonu belirtilen numaralı tablonun adını döndürür.

```
TableName(table_number)
```

TableNumber

Bu kod fonksiyonu belirtilen tablonun numarasını döndürür. İlk tablonun numarası 0 olur.

table_name mevcut değilse NULL döndürülür.

```
TableNumber(table_name)
```

Örnek:

Bu örnekte, yüklenmiş tablolar ve alanlar ile ilgili bilgileri içeren bir tablo oluşturmak istiyoruz.

Önce biraz örnek veri yükleyelim. Bu işlem, bu bölümde açıklanan tablo fonksiyonlarını göstermek için kullanılacak iki tabloyu oluşturur.

```
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 ;
```

Ardından **NoOfTables** fonksiyonunu kullanarak yüklenmiş tablolar aracılığıyla ve sonra da **NoOfFields** fonksiyonunu kullanarak her bir tablodaki alanlar aracılığıyla yineleme yapıyoruz ve tablo fonksiyonlarını kullanarak bilgileri yüklüyoruz.

```
//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;
```

Sonuçta elde edilen Tables tablosu şöyle görünür:

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2

Table	TableNo	TableRows	Field	FieldNo
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

FieldName kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki belirtilen bir sayıya sahip alanın adını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

Söz Dizimi:

```
FieldName(field_number ,table_name)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_number	Referansta bulunmak istediğiniz alanın alan numarası.
table_name	Referansta bulunmak istediğiniz alanı içeren tablo.

Örnek:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

FieldNumber kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki belirtilen bir alanın sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

Söz Dizimi:

```
FieldNumber(field_name ,table_name)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
field_name	Alanın adı.
table_name	Alanı içeren tablonun adı.

field_name alanı table_name içinde yoksa veya table_name mevcut değilse, fonksiyon 0 sonucunu döndürür.

Örnek:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

NoOfFields kod fonksiyonu, daha önce yüklenmiş bir tablo içindeki alanların sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

Söz Dizimi:

```
NoOfFields(table_name)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
table_name	Tablonun adı.

Örnek:

```
LET a = NoOfFields('tab1');
```

NoOfRows

NoOfRows fonksiyonu, daha önce yüklenmiş bir tablo içindeki satırların (kayıtların) sayısını döndürür. Fonksiyon bir **LOAD** deyimi içinde kullanılırsa, geçerli anda yüklenmekte olan tabloya başvuru yapmamalıdır.

Söz Dizimi:

```
NoOfRows(table_name)
```

Bağımsız Değişkenler:

Bağımsız Değişkenler

Bağımsız Değişken	Açıklama
table_name	Tablonun adı.

Örnek:

```
LET a = NoOfRows('tab1');
```

5.27 Trigonometrik ve hiperbolik fonksiyonlar

Bu bölümde, trigonometrik ve hiperbolik işlemleri yapmaya yönelik fonksiyonlar açıklanmaktadır. Fonksiyonların tümünde bağımsız değişkenler, radyan cinsinden hesaplanan açılara çözümlenen ifadelerdir ve burada **x** bir gerçek sayı olarak yorumlanmalıdır.

Tüm açılar radyan cinsinden hesaplanır.

Tüm fonksiyonlar hem veri kod dosyasında hem de grafik ifadelerinde kullanılabilir.

cos

x değerinin kosinüsü. Sonuç -1 ile 1 arasında bir sayıdır.

```
cos( x )
```

acos

x değerinin ters kosinüsü. Fonksiyon yalnızca -1≤**x**≤1 olması durumunda tanımlanır. Sonuç 0 ile π arasında bir sayıdır.

```
acos( x )
```

sin

x değerinin sinüsü. Sonuç -1 ile 1 arasında bir sayıdır.

```
sin( x )
```

asin

x değerinin ters sinüsü. Fonksiyon yalnızca -1≤**x**≤1 olması durumunda tanımlanır. Sonuç - π/2 ile π/2 arasında bir sayıdır.

```
asin( x )
```

tan

x değerinin tanjantı. Sonuç bir gerçek sayıdır.

```
tan( x )
```

atan

x değerinin ters tanjantı. Sonuç - π/2 ile π/2 arasında bir sayıdır.

```
atan( x )
```

atan2

Ters tanjant fonksiyonunun iki boyutlu genelleştirmesi. Başlangıç noktası ile **x** ve **y** koordinatlarının temsil ettiği nokta arasındaki açıyı döndürür. Sonuç - π ile +π arasında bir sayıdır.

```
atan2( y,x )
```

cosh

x değerinin hiperbolik kosinüsü. Sonuç pozitif bir gerçek sayıdır.

```
cosh( x )
```

sinh

x değerinin hiperbolik sinüsü. Sonuç bir gerçek sayıdır.

```
sinh( x )
```

tanh

x değerinin hiperbolik tanjantı. Sonuç bir gerçek sayıdır.

```
tanh( x )
```

acosh

x değerinin ters hiperbolik kosinüsü. Sonuç pozitif bir gerçek sayıdır.

```
acosh( x )
```

asinh

x değerinin ters hiperbolik sinüsü. Sonuç bir gerçek sayıdır.

```
asinh( x )
```

atanh

x değerinin ters hiperbolik tanjantı. Sonuç bir gerçek sayıdır.

```
atanh( x )
```

Örnekler:

Aşağıdaki kod örnek bir tablo yükler ve sonra değerler üzerinde hesaplanan trigonometrik ve hiperbolik işlemleri içeren bir tabloyu yükler.

```
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 Dosya sistemi erişim kısıtlaması

Qlik Sense, güvenlik nedeniyle standart modda veri yükleme komut dosyasındaki yolları ya da dosya sistemini açığa çıkaran fonksiyonları ve değişkenleri desteklemez.

Ancak, dosya sistemi yolları QlikView uygulamasında desteklendiğinden, standart mod devre dışı bırakılabilir ve QlikView komut dosyalarının yeniden kullanılması için eski mod kullanılabilir.

Standart modun devre dışı bırakılması, dosya sistemini açığa çıkararak bir güvenlik riski oluşturabilir.

Standart modu devre dışı bırakma (page 830)

6.1 Dosya tabanlı ODBC ve OLE DB veri bağlantılarına bağlanırken dikkat edilmesi gereken güvenlik unsurları

Dosya tabanlı sürücüleri kullanan ODBC ve OLE DB veri bağlantıları, bağlantı dizgesindeki bağlı veri dosyasının yolunu açığa çıkaracaktır. Yol, veri seçimi diyalog penceresinde veya belirli SQL sorgularında bağlantı düzenlenirken açığa çıkarılabilir. Bu, hem standart mod hem de eski modda görülür.

Veri dosyası yolunun açığa çıkması sorun oluşturuyorsa, mümkünse klasör veri bağlantısı kullanılarak veri dosyasıyla bağlantı kurulması önerilir.

6.2 Standart moddaki kısıtlamalar

Bazı deyimler, değişkenler ve fonksiyonlar standart modda kullanılamaz veya kısıtlamalara sahiptir. Veri kod dosyasında desteklenmeyen deyimlerin kullanılması, kod dosyası çalıştığında bir hataya neden olur. Kod dosyasında hata mesajları bulunabilir. Desteklenmeyen değişkenlerin ve fonksiyonların kullanılması, hata mesajları veya günlük dosyası girişlerine neden olmaz. Bunun yerine, fonksiyon NULL döndürür.

Veri kod dosyasını düzenlediğinizde bir değişken, deyim veya fonksiyonun desteklenmediğini gösteren bir şey yoktur.

Sistem değişkenleri

Sistem değişkenleri

Değişken	Standart mod	Eski mod	Tanım
Floppy	Desteklenmiyor	Desteklenir	Bulunan ilk disket sürücüsünün sürücü harfini döndürür; bu normalde *a:* şeklindedir.

Değişken	Standart mod	Eski mod	Tanım
CD	Desteklenmiyor	Desteklenir	Bulunan ilk CD-ROM sürücüsünün sürücü harfini döndürür. CD-ROM bulunmazsa *c:* döndürülür.
QvPath	Desteklenmiyor	Desteklenir	Qlik Sense yürütülebilir dosyasına yönelik gözatma dizesini döndürür.
QvRoot	Desteklenmiyor	Desteklenir	Qlik Sense yürütülebilir dosyasının kök dizinini döndürür.
QvWorkPath	Desteklenmiyor	Desteklenir	Geçerli Qlik Sense uygulamasına yönelik gözatma dizesini döndürür.
QvWorkRoot	Desteklenmiyor	Desteklenir	Geçerli Qlik Sense uygulamasının kök dizinini döndürür.
WinPath	Desteklenmiyor	Desteklenir	Windows'a yönelik gözatma dizesini döndürür.
WinRoot	Desteklenmiyor	Desteklenir	Windows'un kök dizinini döndürür.
$(include=...)	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**Include/Must_Include** değişkeni, koda eklenmesi ve kod olarak değerlendirilmesi gereken metni içeren bir dosyayı belirtir. Veri eklemek için kullanılmaz. Kodunuzun bölümlerinizi ayrı bir metin dosyasında depolayabilir ve birkaç uygulamada yeniden kullanabilirsiniz. Bu, kullanıcı tanımlı bir değişkendir.

Normal kod deyimleri

Normal kod deyimleri

Deyim	Standart mod	Eski mod	Tanım
Binary	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Başka bir uygulamadan veri yüklemek için **binary** deyimi kullanılır.
Connect	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**CONNECT** deyimi, OLE DB/ODBC arabirimi üzerinden bir genel veritabanına Qlik Sense erişimi tanımlamak için kullanılır. ODBC için, veri kaynağı ilk olarak ODBC yöneticisi kullanılarak belirlenmelidir.
Directory	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**Directory** deyimi, yeni bir **Directory** deyimi oluşturulana dek sonraki **LOAD** deyimlerinde hangi dizinde veri dosyaları aranacağını belirler.
Execute	Desteklenmiyor	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**Execute** deyimi, Qlik Sense verileri yüklediği sırada diğer programları çalıştırmak için kullanılır. Örneğin, gerekli olan dönüştürmeleri yapmak için.

Deyim	Standart mod	Eski mod	Tanım
LOAD from ...	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**LOAD** deyimi, alanları bir dosyadan, kod içinde tanımlanmış verilerden, daha önceden yüklenmiş tablodan, web sayfasından, ardından gelen **SELECT** deyiminin sonucundan veya verileri otomatik olarak oluşturarak yükler.
Store into ...	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**Store** deyimi bir QVD, CSV veya text dosyası olşuşturur.

Kod kontrol ifadeleri

Kod kontrol ifadeleri

Deyim	Standart mod	Eski mod	Tanım
For each... filelist mask/dirlist mask	Desteklenen giriş: Kitaplık bağlantısını kullanan yol Döndürülen çıktı: Kitaplık bağlantısı	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol Döndürülen çıktı: Girişe bağlı olarak kitaplık bağlantısı veya dosya sistemi yolu	filelist mask söz dizimi, geçerli dizinde olup **filelist mask** ile eşleşen tüm dosyaların virgülle ayrılmış bir listesini oluşturur. **dirlist mask** söz dizimi, geçerli dizinde olup dizin adı maskesiyle eşleşen tüm dizinlerin virgülle ayrılmış bir listesini oluşturur.

Dosya fonksiyonları

Dosya fonksiyonları

Fonksiyon	Standart mod	Eski mod	Tanım
Attribute()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Farklı medya dosyalarının meta etiketlerinin değerini metin olarak döndürür.

Fonksiyon	Standart mod	Eski mod	Tanım
ConnectString()	Döndürülen çıktı: Kitaplık bağlantı adı	Girişe bağlı olarak kütüphane bağlantı adı veya gerçek bağlantı	ODBC veya OLE DB bağlantıları için etkin bağlantı dizgesini döndürür.
FileDir()	Döndürülen çıktı: Kitaplık bağlantısı	Döndürülen çıktı: Girişe bağlı olarak kitaplık bağlantısı veya dosya sistemi yolu	**FileDir** fonksiyonu, şu anda okunan tablo dosyasının dizinine giden yolu içeren bir dize döndürür.
FilePath()	Döndürülen çıktı: Kitaplık bağlantısı	Döndürülen çıktı: Girişe bağlı olarak kitaplık bağlantısı veya dosya sistemi yolu	**FilePath** fonksiyonu, şu anda okunan tablo dosyasının tam yolunu içeren bir dize döndürür.
FileSize()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**FileSize** fonksiyonu, filename dosyasının veya filename belirtilmemişse, şu anda okunan tablo dosyasının bayt cinsinden boyutunu içeren bir tamsayı döndürür.
FileTime()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	**FileTime** fonksiyonu, filename dosyasının son değişikliğinin tarihi ve saati için bir zaman damgası döndürür. Bir filename belirtilmezse, fonksiyon geçerli anda okunan tablo dosyasına başvurur.

Fonksiyon	Standart mod	Eski mod	Tanım
GetFolderPath()	Desteklenmiyor	Döndürülen çıktı: Mutlak yol	**GetFolderPath** fonksiyonu, Microsoft Windows *SHGetFolderPath* fonksiyonunun değerini döndürür. Bu fonksiyon, giriş olarak Microsoft Windows klasörünün adını alır ve klasörün tam yolunu döndürür.
QvdCreateTime()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Bu kod fonksiyonu, bir QVD dosyasından XML üst bilgisi zaman damgasını döndürür (varsa); aksi takdirde NULL döndürür.
QvdFieldName()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Bu kod fonksiyonu, bir QVD dosyasındaki **fieldno** numaralı alanın adını döndürür. Alan yoksa NULL döndürülür.
QvdNoOfFields()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Bu kod fonksiyonu bir QVD dosyasındaki alanların sayısını döndürür.
QvdNoOfRecords()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Bu kod fonksiyonu bir QVD dosyasında o anda bulunan kayıtların sayısını döndürür.
QvdTableName()	Desteklenen giriş: Kitaplık bağlantısını kullanan yol	Desteklenen giriş: Kitaplık bağlantısını veya dosya sistemini kullanan yol	Bu kod fonksiyonu bir QVD dosyasında depolanan tablonun adını döndürür.

Sistem fonksiyonları

Sistem fonksiyonları

Fonksiyon	Standart mod	Eski mod	Tanım
DocumentPath()	Desteklenmiyor	Döndürülen çıktı: Mutlak yol	Bu fonksiyon, geçerli Qlik Sense uygulamasına giden tam yolu içeren bir dize döndürür.
GetRegistryString()	Desteklenmiyor	Desteklenir	Verilen kayıt defteri yoluyla adlandırılan kayıt defteri anahtarının değerini döndürür. Bu fonksiyon, grafik ve kodda benzer şekilde kullanılabilir.

6.3 Standart modu devre dışı bırakma

Mutlak veya göreli dosya yollarına ve kütüphane bağlantılarına başvuran QlikView kod dosyalarını yeniden kullanmak için standart modu devre dışı bırakabilir, başka bir deyişle eski modu ayarlayabilirsiniz.

Standart modun devre dışı bırakılması, dosya sistemini açığa çıkararak bir güvenlik riski oluşturabilir.

Qlik Sense

Qlik Sense için standart mod, **Standart mod** özelliği kullanılarak QMC içinde devre dışı bırakılabilir.

Qlik Sense Desktop

Qlik Sense Desktop ürününde, standart/eski modu *Settings.ini*'de ayarlayabilirsiniz.

Qlik Sense Desktop uygulamasını varsayılan yükleme konumunu kullanarak yüklediyseniz, *Settings.ini* dosyası *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini* konumunda olur. Qlik Sense Desktop uygulamasını kendi seçtiğiniz bir klasöre yüklediyseniz, *Settings.ini* dosyası yükleme yolunun *Engine* klasöründe olur.

Aşağıdakileri yapın:

1. Metin düzenleyicisinde *Settings.ini* dosyasını açın.
2. *StandardReload=1* değerini *StandardReload=0* olarak değiştirin.
3. Dosyayı kaydedin ve Qlik Sense Desktop uygulamasını başlatın.

Qlik Sense Desktop artık eski modda çalışır.

Ayarlar

StandardReload için kullanılabilen ayarlar şunlardır:

- 1 (standart mod)
- 0 (eski mod)

7 Qlik Sense içinde desteklenmeyen QlikView fonksiyonları ve deyimleri

QlikView kod dosyalarında ve grafik ifadelerinde kullanılabilen çoğu fonksiyon ve deyim, Qlik Sense içinde de desteklenir, ancak aşağıda açıklanan istisnalar da görülebilir.

7.1 Qlik Sense içinde desteklenmeyen kod deyimleri

Qlik Sense içinde desteklenmeyen QlikView kod deyimleri

Deyim	Yorumlar
Command	Bunun yerine **SQL** kullanın.
InputField	

7.2 Qlik Sense içinde desteklenmeyen fonksiyonlar

Bu liste, Qlik Sense içinde desteklenmeyen QlikView kod ve grafik fonksiyonlarını açıklamaktadır.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Qlik Sense içinde desteklenmeyen önekler

Bu liste, Qlik Sense içinde desteklenmeyen QlikView öneklerini açıklamaktadır.

- **Bundle**
- **Image_Size**
- **Info**

8 uygulamasında tavsiye edilmeyen fonksiyonlar ve deyimlerQlik Sense

QlikView kod dosyalarında ve grafik ifadelerinde kullanılan çoğu fonksiyon ve deyim Qlik Sense uygulamasında da desteklenir, ancak bazılarının Qlik Sense uygulamasında kullanılması tavsiye edilmez. Önceki Qlik Sense sürümlerinde bulunan ancak kullanımdan kaldırılmış olan işlev ve deyimler de mevcuttur.

Uyumluluk nedenleriyle amaçlarına uygun bir şekilde çalışacaklardır, ancak gelecekteki sürümlerde kaldırılabileceği için kodun bu bölümdeki tavsiyelere göre güncellenmesi tavsiye edilir.

8.1 Qlik Sense uygulamasında tavsiye edilmeyen kod deyimleri

Bu tabloda, Qlik Sense uygulamasında kullanılması tavsiye edilmeyen kod deyimleri yer almaktadır.

Tavsiye edilmeyen kod deyimleri

Deyim	Tavsiye
Command	Bunun yerine **SQL** kullanın.
CustomConnect	Bunun yerine **Custom Connect** kullanın.

8.2 Qlik Sense uygulamasında tavsiye edilmeyen kod deyimi parametreleri

Bu tabloda, Qlik Sense uygulamasında kullanılması tavsiye edilmeyen kod deyimi parametreleri açıklanmaktadır.

Tavsiye edilmeyen kod deyimi parametreleri

Deyim	Parametreler
Buffer	Şunun yerine **Incremental** kullanın: • **Inc** (önerilmez) • **Incr** (önerilmez)

Deyim	Parametreler
LOAD	Aşağıdaki parametre anahtar sözcükleri, QlikView dosya dönüştürme sihirbazları tarafından oluşturulur. Veriler yeniden yüklendiğinde işlev korunur, ancak Qlik Sense şu parametrelerle deyim oluşturmak için kılavuzlu destek/sihirbazlar sağlamaz: • 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 Qlik Sense uygulamasında tavsiye edilmeyen fonksiyonlar

Bu tabloda, Qlik Sense uygulamasında kullanılması tavsiye edilmeyen kod ve grafik fonksiyonları açıklanmaktadır.

Tavsiye edilmeyen fonksiyonlar

Fonksiyon	Tavsiye
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Bunun yerine Aralık fonksiyonlarını kullanın. *Aralık fonksiyonları (page 702)*
Color() **QliktechBlue** **QliktechGray**	Bunun yerine diğer renk fonksiyonlarını kullanın. Aynı renkleri elde etmek için **QliktechBlue()**, **RGB(8, 18, 90)** ile ve **QliktechGray** ise **RGB(158, 148, 137)** ile değiştirilebilir. *Renk fonksiyonları (page 415)*
QlikViewVersion	Bunun yerine **EngineVersion** kullanın. *EngineVersion (page 816)*
ProductVersion	Bunun yerine **EngineVersion** kullanın. *EngineVersion (page 816)*
QVUser	
Year2Date	Bunun yerine **YearToDate** kullanın.
Vrank	Bunun yerine **Rank** kullanın.
WildMatch5	Bunun yerine **WildMatch** kullanın.

ALL niteleyicisi

QlikView uygulamasında, **ALL** niteleyicisi bir ifadeden önce gelebilir. Bu, **{1} TOTAL** kullanmakla eşdeğerdir. Bu tür bir durumda, hesaplama, grafik boyutları ve geçerli seçimler göz ardı edilerek, belgedeki alanında tüm değerleri üzerinden yapılır. Belgedeki mantıksal durumdan bağımsız olarak, her zaman aynı değer döndürülür. **ALL** niteleyicisi kullanılırsa, **ALL** niteleyicisi bir kümeyi kendi başına tanımladığı için bir set ifadesi kullanılamaz. Eski sürümlerle uyumluluk nedeniyle, **ALL** niteleyicisi bu Qlik Sense sürümünde çalışmaya devam eder; ancak sonraki sürümlerde kaldırılabilir.