

----- **Pick() , Ceil() ,** -----

Pick(Ceil(Rand()*4), 'Received', 'Approved', 'Pending', 'Denied')

//Ceil avrundar värdet till hösta heltal

// pick väljer en av värdena 'Received', 'Approved', 'Pending', 'Denied' beroende på första indata (Ceil....)

----- **FieldValue () , Peek()** -----

FieldValue(field_name , elem_no)

FieldValue('First name','1') Finds value of row number '1' in the field 'First Name'

Peek(field_name[, row_no[, table_name]])

Peek() finds the value of a field in a table for a row that has already been loaded or that exists in internal memory.

----- **Mapping** -----

Map:

MAPPING LOAD * INLINE [

ID, Status

1,Received

2,Approved

3,Pending

4,Denied

];

Data:

LOAD

ApplyMap('Map',Ceil(Rand()*4)) AS Status

Autogenerate xx;

----- **LookUp** -----

Lookup('Category', 'ProductID', ProductID, 'ProductList')

Lookup(1, 2, 3, 4)

1. Värdena som ska fyllas in i kolumnen (från en annan tabell till denna tabell)
2. Värdena som ska matchas till (Från en annan tabell)
3. Värdena som punkt 2 ska matchas till (samma tabell som man fyller i)
4. Tabellen som värdena i punkt 1 ska tas

----- **IterNo(), RecNo(), RowNo()** -----

IterNo() används som räknare inom while loopar

RowNo() ger radnummer

RecNo() används som räknare för Autogenerate

#TempTest:

load * inline [

FIELD

```
one
two
three
];
```

```
FOR Each a in FieldValueList('FIELD')
Test:
LOAD
'$ (a)' & '-' & RecNo() as NEWFIELD,
'$ (a)' & '-' & RowNo() as NEWFIELD2 ,
'$ (a)' & '-' & IterNo() as NEWFIELD3
AutoGenerate 2
while IterNo() < 4;
NEXT a
```

```
Drop table #TempTest;
```

----- sum({1}sales) vs Total(Sum(sales)) -----

```
=====
Sales:
Load * Inline
[
  Customer, Sales, Brand
  A, 100, B1
  B, 120, B2
  C, 90, B1
  D, 110, B2
];
=====
```

Use Text objects to understand the logic...

=SUM(Sales)

The above expression gives you 420 but it will change according to your selection on Customer or Brand.

=SUM({1}Sales)

The above expression gives you 420 but it **will not change according to your selection** on Customer or Brand.

So the answer would be 420 even after selecting any dimension

=SUM(Total Sales)

The above expression will give you Total Sales **ignoring dimension** but if you select any dimension, it will change accordingly. SUM(Total Sales) is useful if you want to show Total Sales against each line in Pivot or Straight Table or in any other objects.

Create a Pivot Table

Dimensions

Customer

Brand

Expressions

SUM(Total Sales)

SUM(Total <Brand> Sales)

Here second expression will give you Total Sales Brand wise....

=SUM(All Sales)

Same as SUM({1}Sales).....

Sum({1}TOTAL [Premie]) *Detta visar totala värden I hela data*

Sum({<[Försäkringsbolag]>} ALL [Premie])

Sum({1} [Premie])

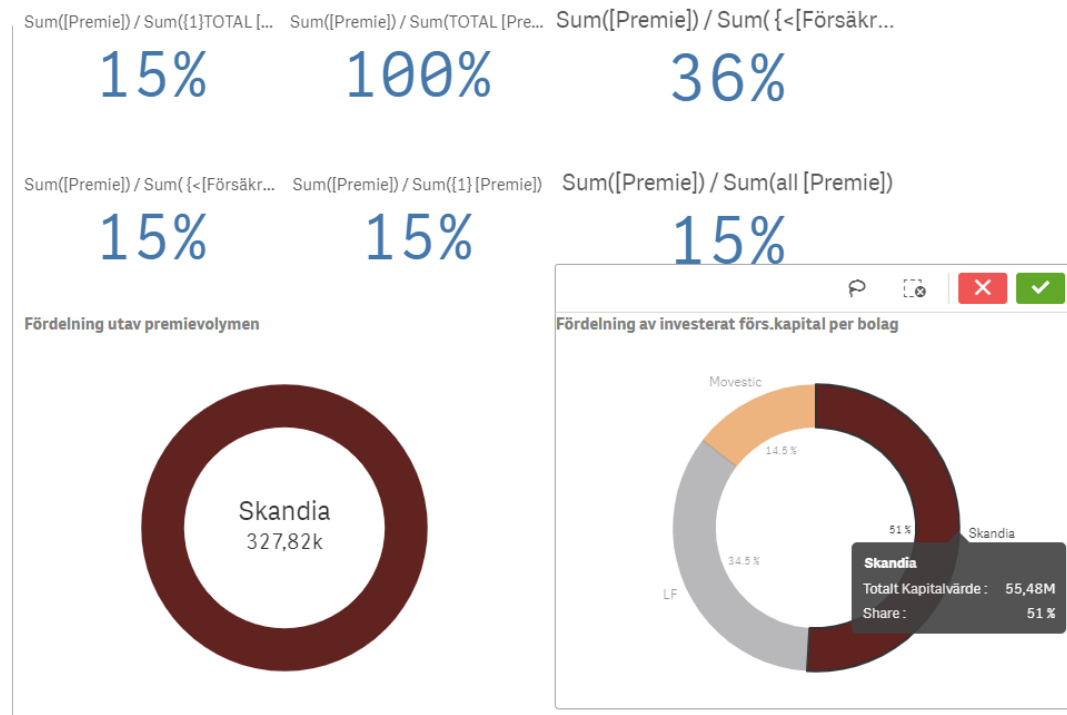
Sum(all [Premie])

Sum(TOTAL [Premie])

Den ändras med slicersval

Sum({<[Försäkringsbolag]>} TOTAL [Premie])

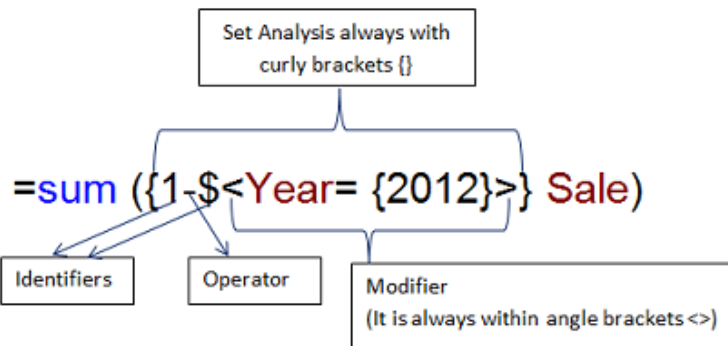
Den ändras med alla slicersval men inte med {<[Försäkringsbolag]>} Slicer.



Example	Result
sum({\$<OrderDate = DeliveryDate>} Sales)	Returns the sales for the current selection where OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Returns the sales for region US, disregarding the current selection.
sum({\$<Region = >} Sales)	Returns the sales for the selection, but with the selection in Region removed.
sum({<Region = >} Sales)	Returns the same as the example above. When the set to modify is omitted, \$ is assumed.
sum({\$<Year={2000}, Region={U*}>} Sales)	Returns the sales for the current selection, but with new selections both in Year and in Region.

<https://help.qlik.com/en-US/qlikview/12.0/Subsystems/Client/Content/ChartFunctions/SetAnalysis/set-analysis-expressions.htm>

- Identifier
- Operator
- Modifier



Expressions	Results
=sum ({1} Sale)	Return total sales of the application irrespective of selection, it will not disregard dimensions.
=sum ({\$} Sale)	Return sales for current selection
=sum ({\$1} Sale)	Return the sale of previous selection

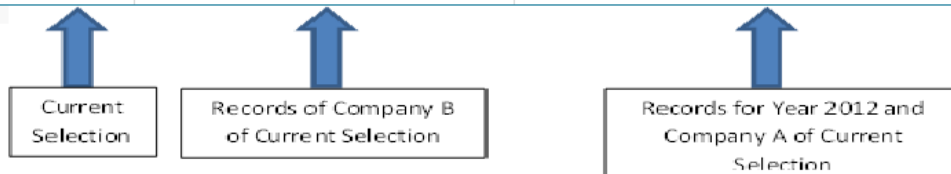
The Modifier in <> overwrite the first identifier:

Expressions	Results
=Sum ({<Company_Name= {'A'}>} Sale)	Return sales of Company A of current selection.
=sum ({1<Company_Name= {'A'}>} Sale)	Return sales of Company A irrespective of selection
=sum ({1-\$<Company_Name= {'A'}>} Sale)	Return sales of Company A excluding current selection
=Sum ({<\$<Year= {2012}, Company_Name= {'A'}>} Sale)	Return sales of Company A of year 2012
=sum ({<\$<Company_Name= {'A','B'}>} Sale)	Return sales of Company A and B for Current selection
=sum ({<\$ <Year= >} Sale)	Return sales for all three years of current selection

Year

2011 2012 2013

Company_Name	Sum({\$}Sale)	=sum({<\$<Company_Name= {'B'}>}Sale)	=sum({<\$<Year= {2012}, Company_Name= {'A'}>}Sale)
A	1000	0	1000
B	1000	1000	0
C	1000	0	0



Dollar Sign Expansion:

If we want to compare current year sale with previous year, previous year sales should reflect values in relation to current selection of year. For example if current selection of year is 2012, previous year should be 2011 and for current selection of year 2013, previous year is 2012.

“=Sum ({<\$<Year = {\$ (=Max (Year)-1)} >} Sale) “

Above expression always returns sale for previous year. Here \$ sign (Font color red) is used to evaluate the value for previous year. \$ sign is used to evaluate expression and to use variables in set modifiers. If we have variable that holds last year value (**vLASTYEAR**) then expression can be written as:

“=Sum ({<\$vLASTYEAR)} >} Sale) “

Indirect SET ANALYSIS: Function P() and E()

Let us take a scenario, where we want to show current sales of the companies who had sales last year.

Expression should be similar like:

```
=sum({$<Year={$(=Max(Year))},Company_Name={Companies who had sales last year}> } Sale)
```

First we have to identify companies who had sales last year. To fix this problem, we will use function P() that is used to identify values within a field and function E() that exclude values within a field.

Expressions	Result
Company_Name=p(Company_Name)	All companies who had sales across year (2011 to 2013)
Company_Name=P({<Year={2012}>} Company_Name)>}}	All companies who had sales in year (2012)
Company_Name= P({<Year={\$(=Max(Year)-1)}>} Company_Name)	All companies who had sales in previous year

Finally, we have expression:

```
=sum({$<Year={$(=Max(Year))},Company_Name=P({<Year={$(=Max(Year)-1)}>}Company_Name)>}Sale)
```

<https://www.analyticsvidhya.com/blog/2014/01/set-analysis-qlikview/>

----- SUM(Total Value) Aggr(nodistinct) -----

TempTest:

load * inline [

ColA, ColB, Value

A,	a,	200
A,	b,	250
B,	a,	300
A,	b,	450
C,	b,	400
C,	c,	500

];

ColA	ColB	Value	=sum(Value)	=Sum(Total <ColB> Value)	=Sum(Total Value)	=Aggr(sum(Value), ColB)	=Aggr(Nodistinct sum(Value), ColB)
Totals			2100	2100	2100	-	-
A	b	450	450	1100	2100	-	1100
A	b	250	250	1100	2100	1100	1100
A	a	200	200	500	2100	500	500
B	a	300	300	500	2100	-	500
C	c	500	500	500	2100	500	500
C	b	400	400	1100	2100	-	1100

=Aggr(sum(Value), ColA)	=Aggr(Nodistinct sum(Value), ColA)	=Aggr({<ColB ={'b','a'}, ColA= -{'b'} >} nodistinct sum(Value), ColA, ColB)
-	-	-
-	900	700
-	900	700
900	900	200
300	300	-
-	900	-
900	900	400

Sum({<[1]>} TOTAL <2> [Premie])

{<[1] ={'x'}}>} summan Filtreras med x dvs bara dem som innehåller x får summan av Premie
Om man skriver {<[1]>} eller {<[1]=>} istället då används för alla värde oavsett slicervar.

TOTAL gör att summan görs av hela kolumnen (grupperingar tas bort) och <2> gör gruppering enligt värdena i kolumnen (t.ex.).

Above(total x) kan skjuta x ett steg neråt. Utan total ska grupperingen påverka det. Om man skriver Above(total x,1,3) då detta ger tillbaka tre värde ovan samma rad.

ColA	Q	ColB	Q	Value	Q	=Aggr(Nodistinct sum([1] Value), ColA, ColB)	=Aggr(sum(Value), ColB)	=Aggr(Nodistinct sum(Value), ColB)	=Aggr(sum(Value), ColA)	=Aggr(Nodistinct sum(Value), ColA)	=Aggr([<ColB =['b','a'], ColA =['b'] >) nodistinct sum(Value), ColA, ColB)
Totals						-	-	-	-	-	-
A	a			200		200	850	850	1450	1450	200
A	b			250		700	1420	1420	-	1450	700
A	b			450		700	-	1420	-	1450	700
A	c			550		550	2170	2170	-	1450	-
B	a			100		250	-	850	1150	1150	-
B	a			150		250	-	850	-	1150	-
B	b			300		300	-	1420	-	1150	-
B	c			600		600	-	2170	-	1150	-
C	a			400		400	-	850	1840	1840	400
C	b			420		420	-	1420	-	1840	420
C	c			500		1020	-	2170	-	1840	-
C	c			520		1020	-	2170	-	1840	-

ColA	Q	ColB	Q	Value	Q	=sum(Value)	=Aggr(Nodistinct sum([1] Value), ColA, ColB)	=([Aggr(nodistinct above(total sum([1] Value)),ColA,ColB)])	= above(Total sum(Total <ColA,ColB> Value))	=Sum(Total <ColB> Value)	=Sum(Total Value)
Totals						4440	-	-	-	4440	4440
A	a			200		200	200	-	-	850	4440
A	b			250		250	700	200	200	1420	4440
A	b			450		450	700	200	700	1420	4440
A	c			550		550	550	700	700	2170	4440
B	a			100		100	250	550	550	850	4440
B	a			150		150	250	550	250	850	4440
B	b			300		300	300	250	250	1420	4440
B	c			600		600	600	300	300	2170	4440
C	a			400		400	400	600	600	850	4440
C	b			420		420	420	400	400	1420	4440
C	c			500		500	1020	420	420	2170	4440
C	c			520		520	1020	420	1020	2170	4440

2 conditions within 1 expression

=COUNT ({< UDATE = {'>= \$(=Date(vStartDate))<= \$(=Date(vEndDate))' } , SCORECARDNUMBER = {'>= \$(=ScorecardStart)<= \$(=ScoreCardEnd)' } >} DOCUMENT_COUNT)

ColA	Q	ColB	Q	Value	Q	=AVG(Total <ColA> Value)	=Stdev(Total <ColA> Value)	=Stdev({<ColB = -['b']>} Total <ColB> Value)	=Stdev(Total <ColB> Value)
Totals						370	166,24188	198,33233	166,24188
A	a			200		362,5	165,2019	131,49778	131,49778
A	b			250		362,5	165,2019	-	95,39392
A	b			450		362,5	165,2019	-	95,39392
A	c			550		362,5	165,2019	43,493295	43,493295
B	a			100		287,5	225	131,49778	131,49778
B	a			150		287,5	225	131,49778	131,49778
B	b			300		287,5	225	-	95,39392
B	c			600		287,5	225	43,493295	43,493295
C	a			400		460	58,878406	131,49778	131,49778
C	b			420		460	58,878406	-	95,39392
C	c			500		460	58,878406	43,493295	43,493295
C	c			520		460	58,878406	43,493295	43,493295

=Sum({<ColA =["A"]>} Value)

1,45k

aggregated standard deviation

=stdev(aggr(stdev(Value),ColA))

84,14

https://help.qlik.com/en-US/sense/September2018/Subsystems/Hub/Content/Sense_Hub/ChartFunctions/ColorFunctions/color-functions-charts.htm

=Colormix1 ((Value/ MAX(Total Value)) , RGB (255, 150, 100) , RGB (100, 150, 255))

=Colormix2 ((Value/ MAX(Total Value)-0.5)*2 ,RGB (255, 100, 0) , RGB (0, 150, 100),RGB (0, 0, 0))

//=ColorMapJet (((Value-Min(Total Value)+0.01)/Max(Total Value))))

Colorized each dimension in the pivot table::::

=IF(Dimensionality()= 1

, RGB (250,250,230) //Yellow

,IF(Dimensionality()= 2

,RGB(230,250,230)// Green

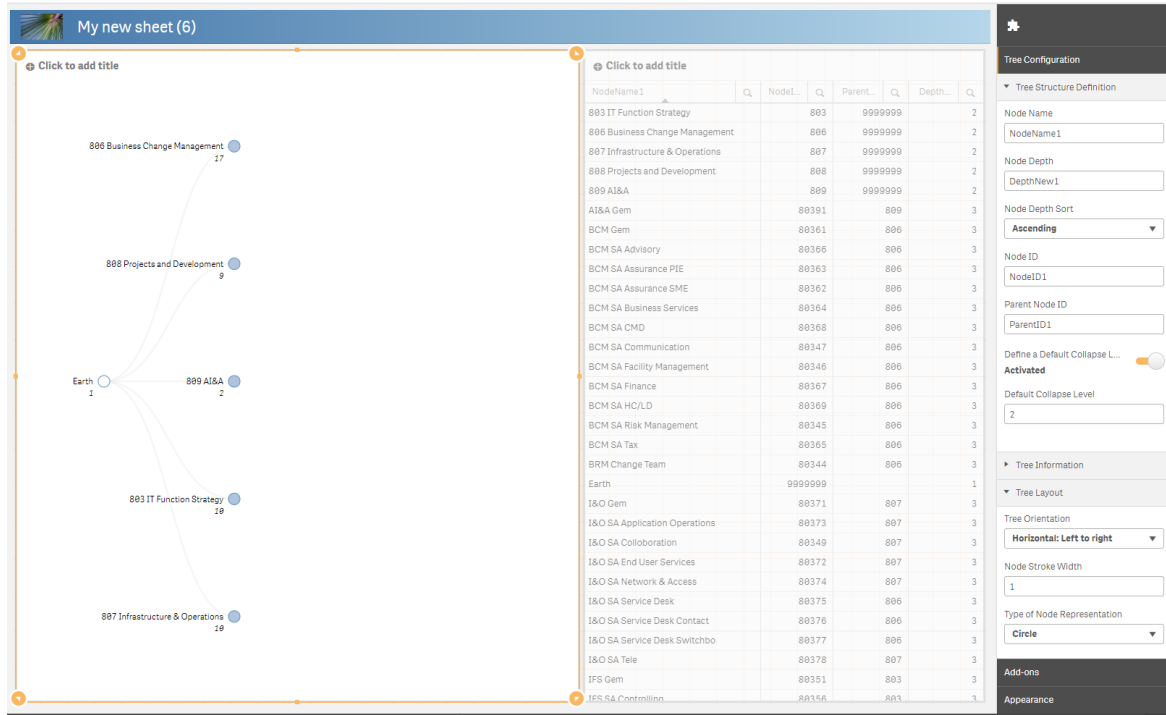
,IF(Dimensionality()= 3

, RGB(230,250,250) //Blue

, RGB (250,230,230) //Red

)))

----- Hierarchy -----



Test4:

Load distinct

```
Num#([Kostnadsställe]) AS NodeID1,
Num#(left("Function Area",3)) AS ParentID1, / Num#(left([Kostnadsställe],3)) AS ParentID1,
[KostnadsställeNamn] AS NodeName1
```

Resident DimOrganisation;

Concatenate(Test4)

Load

```
Num#(left([Function Area],3)) AS NodeID1,
9999999 AS ParentID1,
[Function Area] AS NodeName1
```

RESIDENT DimOrganisation;

Concatenate(Test4)

LOAD * inline

```
[
NodeID1, ParentID1, NodeName1
9999999, , Earth
];
```

Hierarchy (NodeID1, ParentID1, NodeName1, ParentName1, NodeName1, PathName1, '\', DepthNew1)

Load * Resident Test4;

autonumber(expression[, AutoID])

This script function returns a unique integer value for each distinct evaluated value of *expression*, The expression can be composite from some fields. (field1&field2....)

Hierarchy:

```
Hierarchy(BolagsID, ParentID, Bolagsnamn, Parent, Bolagsnamn, PathName, '\', Depth)
LOAD
```

```

Bolagsnamn,
AutoNumber(Bolagsnamn) as BolagsID,
if(Ägarbolag <> 'Koncernmoder',AutoNumber(Ägarbolag)) as ParentID
//AutoNumber(Ägarbolag) as ParentID
FROM [lib://30.2.TAX/8.Import\Uppsalavisualisering.xlsx]
(ooxml, embedded labels, header is 1 lines, table is [Qlikförteckning Bolag])
where len(trim(Bolagsnamn)) > 0 ;

```

----- vissa Definition -----

variabelnamn, definition

"BU" "Affärsenhet"

"CR", "Client responsible, kundansvarig"

"Intäkt (R12)", "Upparbetat värde senaste 12 månader"

"Marknadspenetration", "Andel företag/koncerner som är PwC-kunder av alla företag/koncerner"

"Omsättning", " Med omsättning avses ett företags eller en organisations totala försäljning (såväl kontant som fakturerad) under en viss tidsperiod, vanligen per år."

"Proposition", "Beskrivning av affärens område"

"Prospect", "Företag på marknaden där varken upparbetade intäkter eller affärsmöjligheter har registrerats under de senaste 12 månaderna"

"Segment (bolag)", "Sätts utifrån företagets nettoomsättning enligt CMD-specifik klassificering"

"Segment (koncern)", "Sätts utifrån koncernens nettoomsättning enligt CMD-specifik klassificering"

"Target", "Ett företag där aktiv bearbetning pågår och affärsmöjlighet finns registrerad"

"Tier (bolag)", "Sätts utifrån företagets nettoomsättning enligt CMD-specifik klassificering"

"Tier (koncern)", "Sätts utifrån koncernens nettoomsättning enligt CMD-specifik klassificering"

"Uppskattat värde", "Säljarens uppskattning av affärens värde ("Estimated value")"

"Viktat värde", "Ett värde beräknat från uppskattad affärsvärde och vilken fas försäljningen befinner sig i ("weighted value")"

----- For, Next loop -----

```

For i= NoOfTables()-1 to 0 step -1
    LET vTable = TableName$(i);

    IF WildMatch('$(vTable)', 'Data*') THEN
        LEFT JOIN ([Fact]) LOAD * RESIDENT [$(vTable)];
        DROP TABLE [$(vTable)];
    ENDIF

```

Next i

----- To ignore Excel Header -----

```

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

```

Using the **header is 1 lines** specifier, the first line will not be loaded as data. In the example, the **embedded labels** specifier tells Qlik Sense to interpret the first non-excluded line as containing field labels.

```

LOAD Col1, Col2

```

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

----- rangesum(above(sum(Field),0,3))-----

<https://community.qlik.com/docs/DOC-4252>

`Aggr(Above(Sum(Sales)),Year,Month)`

displays the value from the previous month from the same year. But if you change the order of the dimensions, as in

`Aggr(Above(Sum(Sales)),Month,Year)`

the expression will display the value from the same month from the previous year. The only difference is the order of the dimensions. The latter expression is sorted first by Month, then by Year. The result can be seen below:

Sum(Sales)				
Year	Month	Sum(Sales)	Only(Aggr(Above(total Sum({\$<Year=,Month=>}Sales)),Year,Month))	Only(Aggr(Above(Sum({\$<Year=,Month=>}Sales)),Month,Year))
2012	Jan	783	-	-
2012	Feb	676		783
2012	Mar	547		676
2012	Apr	753		547
2012	May	587		753
2012	Jun	786		587
2012	Jul	915		786
2012	Aug	992		915
2012	Sep	954		992
2012	Oct	1018		954
2012	Nov	969		1018
2012	Dec	1087		969
2013	Jan	878		1087
2013	Feb	785		878
2013	Mar	788		785
2013	Apr	828		788
2013	May	770		828

An Aggr() table is always sorted by the load order of the dimensions, one by one. This means that you can change the meaning of Above() by changing the order of the dimensions. With this, I hope that you understand the Above() function better.