Nested Aggregations and Related Issues

This chapter exemplifies some important techniques in relation to nested aggregations and the use of the ***Advanced Aggregation*** function in charts.

Note!
As of QlikView version 9 no more than 100 levels of nesting is allowed.

Nested Aggregations with Total Qualifier

As a general rule, it is not allowed to nest aggregations in a QlikView chart expression. From version 7.5 there is however one very important exception to this rule. As long as you use the **total** qualifier in the inner aggregation function, the nesting is allowed.

Say for example that you want to calculate the sum of the field Sales, but only include transactions with an OrderDate equal to the last year. The last year can be obtained via the aggregation function **max(total** year(OrderDate**))**.

An aggregation as follows would then do the job:

```
sum( if(year(OrderDate)=max(total year(OrderDate)), Sales)).
```

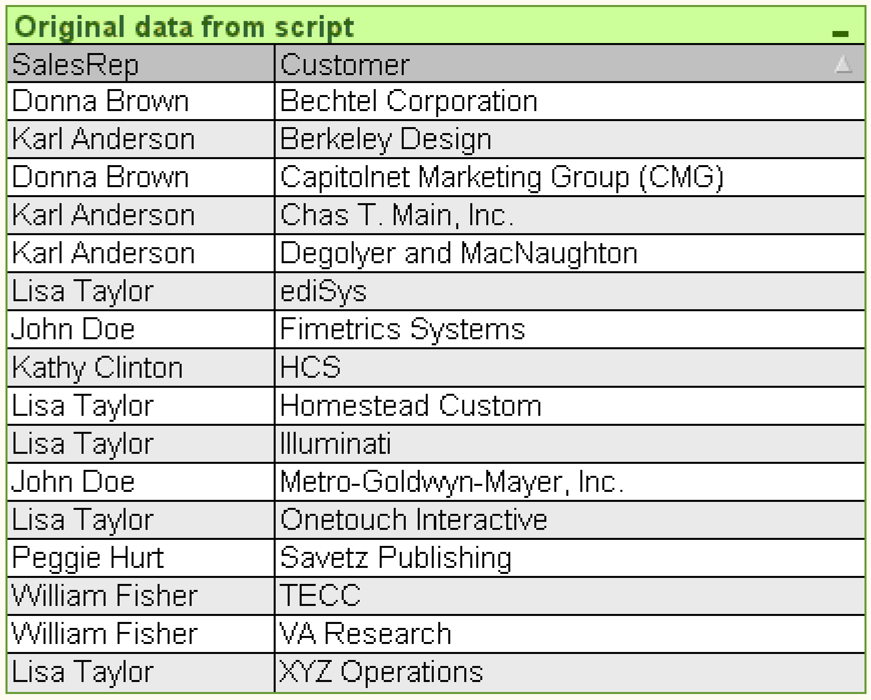
The inclusion of the **total** qualifier is absolutely necessary for this kind of nesting to be accepted by QlikView, but then again also necessary for the desired comparison. This type of nesting need is quite common and should be used wherever suitable.

Nested Aggregations with the Aggregation Function

Nesting with **total** is not always enough. For more generic nesting capabilities you will have to use the ***Advanced Aggregation*** function in combination with calculated dimensions, see ***Add calculated dimension...*** .

Example:

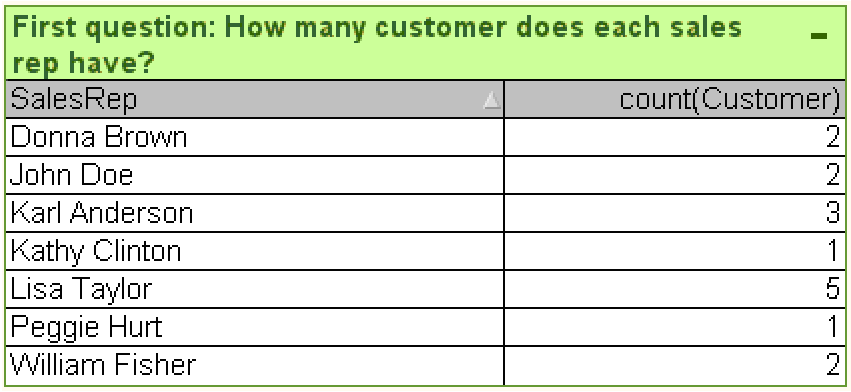
The following data has been read from the script:



Original data from script

SalesRep	Customer
Donna Brown	Bechtel Corporation
Karl Anderson	Berkeley Design
Donna Brown	Capitolnet Marketing Group (CMG)
Karl Anderson	Chas T. Main, Inc.
Karl Anderson	Degolyer and MacNaughton
Lisa Taylor	ediSys
John Doe	Fimetrics Systems
Kathy Clinton	HCS
Lisa Taylor	Homestead Custom
Lisa Taylor	Illuminati
John Doe	Metro-Goldwyn-Mayer, Inc.
Lisa Taylor	Onetouch Interactive
Peggie Hurt	Savetz Publishing
William Fisher	TECC
William Fisher	VA Research
Lisa Taylor	XYZ Operations

An obvious question given this data would be: "How many customers does each sales rep have?". This is easily done in a standard chart:



First question: How many customer does each sales rep have?

SalesRep	count(Customer)
Donna Brown	2
John Doe	2
Karl Anderson	3
Kathy Clinton	1
Lisa Taylor	5
Peggie Hurt	1
William Fisher	2

Now however, let's ask a couple of new questions based on the knowledge just gained: "How may sales reps have only a single customer? How many have three or more?". If we disregard the fact that you in this simple case can count the numbers in the expression columns by hand, these are types of questions that require a second order of aggregation. The data necessary to make the calculation does not exist in the original fields, nor can it be directly calculated from them.

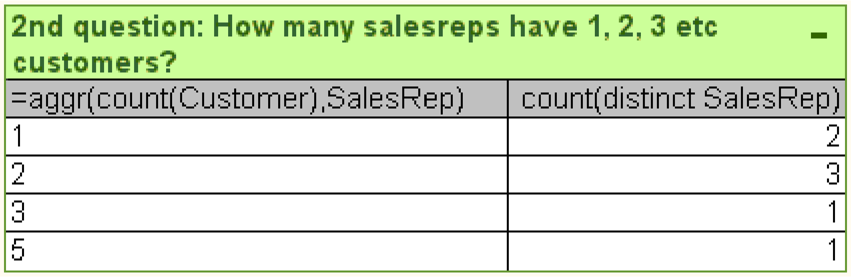
We must simply find a way to use the expression column in the chart above as a dimension in a new chart. The answer lies the *Advanced Aggregation* function. By stating

```
=aggr(count(Customer),SalesRep)
```

as dimension we can essentially perform the calculation of the first chart as an 'inner chart calculation' in a new chart. The new chart could then be given the expression

```
count(distinct SalesRep)
```

and the trick is complete. The **distinct** qualifier is necessary, since QlikView will count the number of lines in the underlying table. The resulting table will look like below:



2nd question: How many salesreps have 1, 2, 3 etc customers?

=aggr(count(Customer),SalesRep)	count(distinct SalesRep)
1	2
2	3
3	1
5	1

Two things are to be noted:

The second chart does in no way require the presence of the first chart. It is fully self-contained with the first order aggregation defined within its dimension.

The possibilities of nesting do not end here. The dimension arguments of the *Advanced Aggregation* function may of course contain calculated dimensions, which in turn make use of the *Advanced Aggregation* function. It should however be relatively easy to loose track of what you are doing when passing the third level of aggregation.

Sum of Rows in Pivot Tables

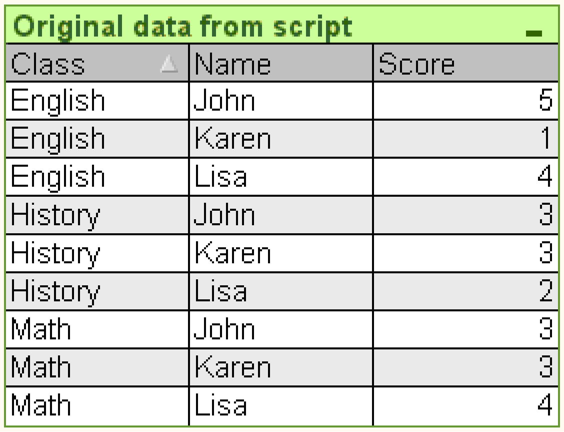
The QlikView straight table has a choice for its totals between a simple sum of rows and a calculated expression total. The QlikView pivot table lacks this choice. Pivot table totals are always calculated as expression total.

This is normally a good thing, since it is a rather rare occasion that a sum of rows total is relevant when the two differ. You should exercise extreme care when using sum of rows on any type of aggregation other than pure sums.

Having issued that warning, we will now look at an example where the sum of rows total is nevertheless the desired result.

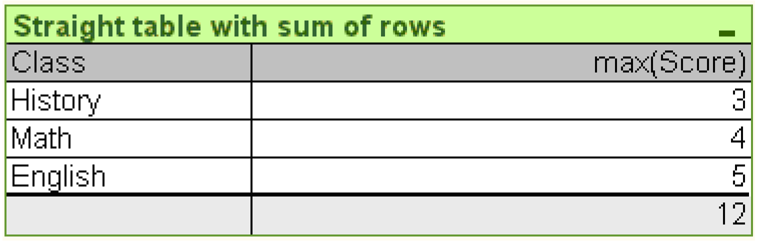
Example:

Let's say that we have a school contest where three person teams get points by their grades in three different classes. The team may select the highest score within the group for each individual class and then add the three top scores together for a total. The following data has been read from the script:


Original data from script

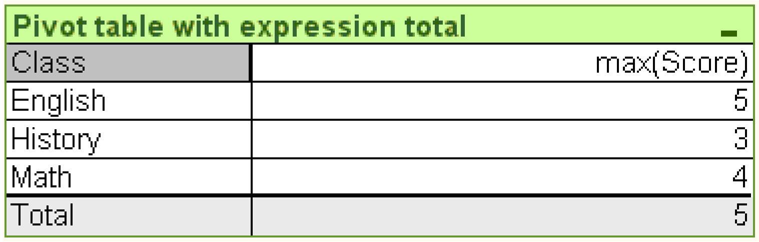
Class	Name	Score
English	John	5
English	Karen	1
English	Lisa	4
History	John	3
History	Karen	3
History	Lisa	2
Math	John	3
Math	Karen	3
Math	Lisa	4

We must now make a chart with Class as dimension and **max(**Score**)** as expression. A straight table with sum of rows will look like follows:


Straight table with sum of rows

Class	max(Score)
History	3
Math	4
English	5
	12

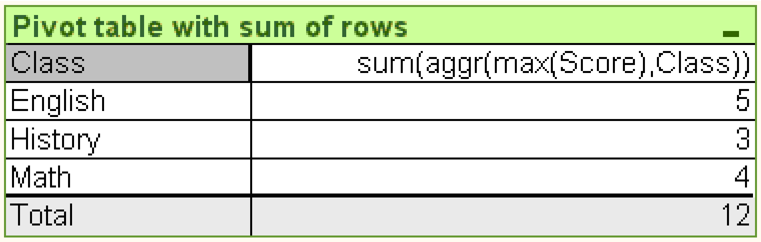
If we for some reason want to display this in a pivot table (not much use here, but if we had more dimensions it may make sense), we run into problems. The straight table above converted into a pivot table would look as follows:


Pivot table with expression total

Class	max(Score)
English	5
History	3
Math	4
Total	5

In this specific case the total of 12 is clearly what we want and 5 is equally wrong for our purposes. Again the **aggr** function comes to our rescue. In this case we use it in the expression, not the dimension.

The original expression is enclosed in an **aggr** function, using the surrounding chart's dimension also as dimension in the **aggr** function. Then we use this bundle as argument to a **sum** aggregation. The result will look like this:


Pivot table with sum of rows

Class	sum(aggr(max(Score),Class))
English	5
History	3
Math	4
Total	12

As you see the total is again the one desired. What happened?

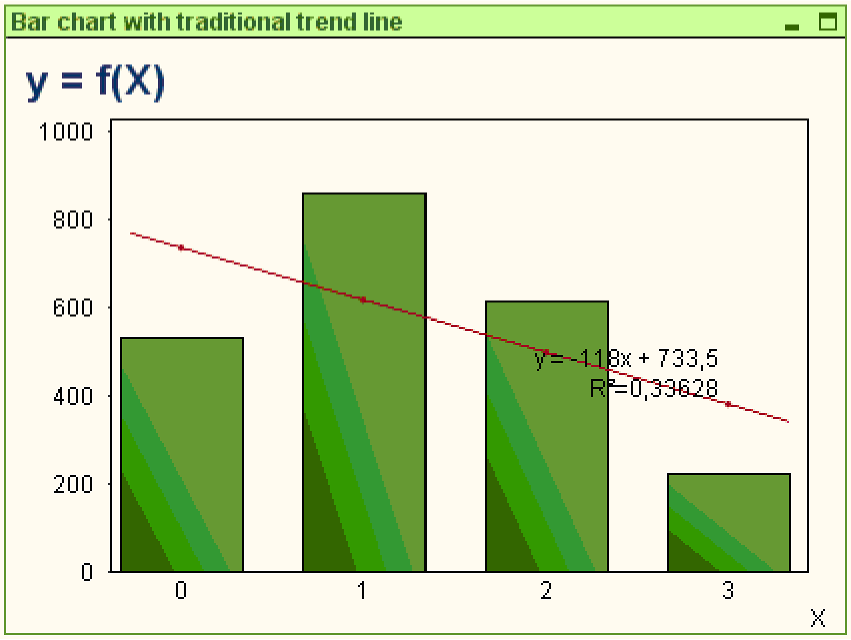
Well the beauty of the ***Advanced Aggregation*** function is that in the individual rows it will evaluate to only a single value. This is because the dimension obviously only has one possible value on each ordinary data row. Since the inner dimension and expression are the same as for the surrounding chart, each value will of course be exactly the same as the result without the enclosing **sum** and ***Advanced Aggregation*** functions.

For the total row however, the *Advanced Aggregation* function will return three values, one for each value of the dimension field. These will in turn be summed by the **sum** aggregation. While formally still being an expression total, the result equals that of sum of rows.

Linear Regression in Table Charts

Linear regression trend lines can be shown in QlikView bitmap charts by means of the **Trendlines** option in the **Expressions** page of **Chart Properties**. It is also possible to display the regression equation.

Example:

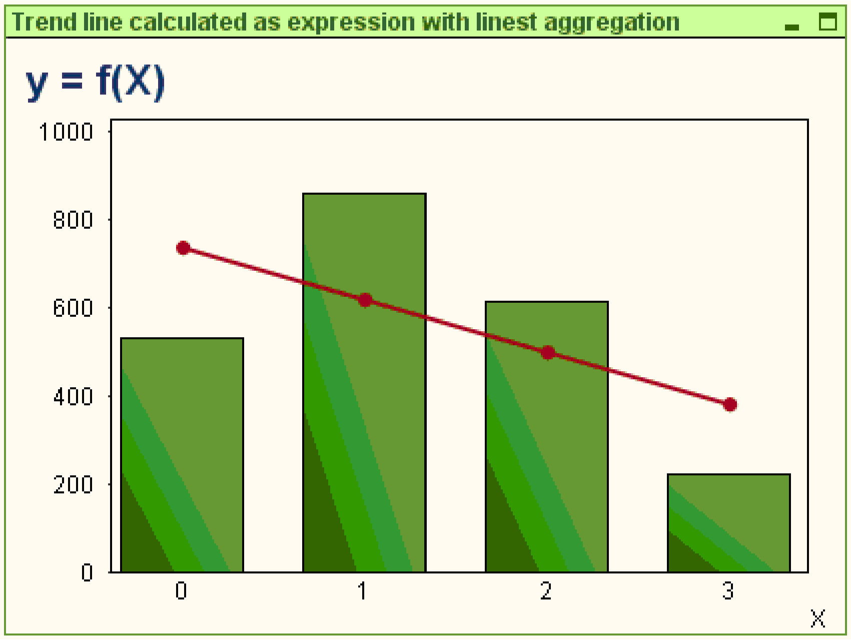


If you want to display the regression data in e.g. a table chart, the regression must be calculated. The *linest_m ([{set_expression}][distinct] [total [<fld {,fld}>]]y-expression, x-expression[, y0 [, x0]])* and *linest_b ([{set_expression}][distinct] [total [<fld {,fld}>]] y-expression, x-expression [, y0 [, x0]])* aggregation functions will give you the required slope and y-intercept values of the linear regression.

To calculate correctly, these functions need to have the entire chart aggregation (expression iterated over dimension) as input. This can be achieved by defining an *Advanced Aggregation* function containing the same base expression and dimension(s) as the containing chart. The *Advanced Aggregation* function is then used as parameters to the **linest** aggregations. The resulting expression could look like follows:

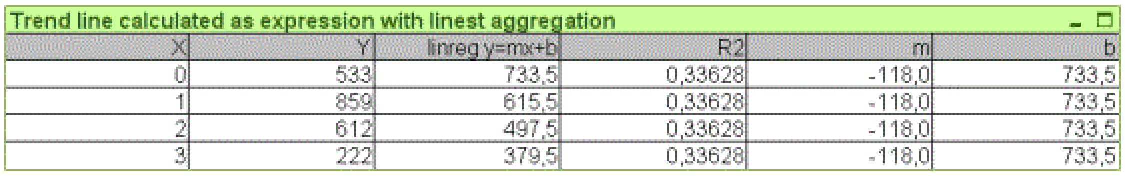
```
linest_m(total aggr(Y,X),X)*X + linest_b(total aggr(Y,X),X)
```

The **only** function is implied around all occurrences of X and Y. The **linest** aggregations should be made with the **total** qualifier, else would the regression parameters be calculated per data point rather than for the whole set of data. The result can be seen in the combo chart below where the regression is shown as a regular line expression.



Note that the trend line here is not a traditional QlikView trend line, but a regular expression plotted as line. You can see the difference from the fact that the expression plot, as opposed to a traditional trend line, is not extrapolated outside the first and last data points.

This chart can converted to a straight table, where the regression values are shown in cells.



Trend line calculated as expression with linest aggregation

X	Y	linreg y=mx+b	R2	m	b
0	533	733,5	0,33628	-118,0	733,5
1	859	615,5	0,33628	-118,0	733,5
2	612	497,5	0,33628	-118,0	733,5
3	222	379,5	0,33628	-118,0	733,5

In the straight table above three extra columns have been added to show the m, b and R^2 values. These of course are constant for all table rows. The expressions needed would look like follows, in order of appearance:

```
linest_r2(total aggr(Y,X),X)
linest_m(total aggr(Y,X),X)
linest_b(total aggr(Y,X),X)
```

QlikView 11.20 SR8