Chart Aggregation functions

These functions can only be used on fields in chart expressions. The argument expression of one aggregation function must not contain another aggregation function.

The aggregation functions are:

If the word **distinct** occurs before an *expression*, duplicates resulting from the evaluation of this *expression* will be disregarded.

If the word **total** occurs before an *expression*, the calculation will be made over all possible values given the current selections, but disregarding the chart dimensions.

The **total** qualifier may be followed by a list of one or more field names within angle brackets. These field names should be a subset of the chart dimensions. In this case the calculation will be made disregarding all chart dimensions except those listed, i.e. one value will be returned for each combination of field values in the listed dimension fields. Also fields which are not currently a dimension in a chart may be included in the list. This may be useful in the case of group dimensions, where the dimension fields are not fixed. Listing all of the dimensions in the group causes the function to work when the cycle or drill-down level changes.

In previous QlikView versions, the **all** qualifier may occur before an *expression*. This is equivalent to using **{1} total**. In such a case the calculation will be made over all the values of the field in the document, disregarding the chart dimensions and current selections. (The same value is always returned regardless of the logical state in the document.) If the **all** qualifier is used, a set expression cannot be used, since the all qualifier defines a set by itself. For legacy reasons, the **all** qualifier will still work in this QlikView version, but may be removed in coming versions.

Expression must not contain aggregation functions, unless these inner aggregations contain the **total** qualifier. For more advanced nested aggregations, please use the ***Advanced Aggregation*** function in combination with calculated dimensions, see ***Add calculated dimension...*** . See also the examples of ***Nested Aggregations and Related Issues***.

By default, the aggregation function will aggregate over the set of possible records defined by the selection. An alternative set of records can be defined by a set expression. See also ***Set Analysis***.

For a better understanding of how to use the aggregate qualifier (total), see ***Examples of Aggregate Qualifiers***.

Basic Aggregation Functions

sum(*[{set_expression}][***distinct** *] [***total** *[<fld {, fld}>]] expression***)**

Returns the aggregated sum of *expression* or *field* iterated over the chart dimension(s).

Examples:

```
sum(Sales)
sum(Price*Quantity)
```

`sum( distinct Price)`

`sum( Sales) / sum( total Sales)` returns the share within the selection

`sum( Sales) / sum( total <Month> Sales)` returns the share within the selection for each *Month*

`sum( Sales) / sum( total <Month, Grp> Sales)` returns the share within the selection for each *Month* and *Grp*

`sum( Sales) / sum( total <Qtr, Month, Week> Sales)` possible syntax for use with a time drill-down group

`sum( {1} total Sales)` returns sales within the entire document

`sum( {BM01} Sales)` returns sales within the selection defined by bookmark BM01

`sum( {$ <Year={2007, 2008}>} Sales)` returns the sales for the current selection but just for the years 2007 and 2008, that is, the same as `sum( if( Year=2007 or Year=2008, Sales))`

min(*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression [, rank])***)**

Returns the numeric minimum value of *expression* or *field* iterated over the chart dimension(s). *Rank* defaults to 1 which corresponds to the lowest value. By specifying *rank* as 2 the second lowest value will be returned. If *rank* is 3 the third lowest value will be returned and so on.

Examples:

```
min( Sales )
min( Sales, 2 )
min( Price*Quantity )
min( total Sales )
min( {1} total Sales )
```

max(*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression [, rank])***)**

Returns the numeric maximum value of *expression* or *field* iterated over the chart dimension(s). *Rank* defaults to 1 which corresponds to the highest value. By specifying *rank* as 2 the second highest value will be returned. If *rank* is 3 the third highest value will be returned and so on.

Examples:

```
max( Sales )
max( Sales, 2 )
max( Price*Quantity )
max( total Sales )
max( {1} total Sales )
```

only(*[{set_expression}][distinct] [total [<fld {, fld}>]] expression***)**

If *expression* or *field* iterated over the chart dimension(s) contain one single value, that value is returned, else NULL is returned. **Only** can return numeric values as well as text values.

Examples:

```
only( Sales )
only( Price*Quantity )
only( total Salesman )
```

mode(*[{set_expression}][distinct] expression***)**

Returns the mode value, i.e. the most commonly occurring value, of *expression* or *field* iterated over the chart dimension(s). If more than one value is equally commonly occurring, NULL is returned. **Mode** can return numeric values as well as text values.

Mode does not support the **total** qualifier.

Examples:

```
mode( Product )
mode( X*Y/3 )
```

firstsortedvalue(*[{set_expression}][**distinct**] [**total** [<fld {, fld}>]] expression [, sort_weight [, n]]***)**

returns the first value of *expression* sorted by corresponding *sort-weight* when *expression* is iterated over the chart dimension(s). *Sort-weight* should return a numeric value where the lowest value will render the corresponding value of *expression* to be sorted first. By preceding the *sort-value* expression with a minus sign, the function will return the last value instead. If more than one value of expression share the same lowest *sort-order*, the function will return null. By stating an n larger than 1, you will get the nth value in order.

Examples:

```
firstsortedvalue ( PurchasedArticle, OrderDate )
firstsortedvalue ( PurchasedArticle, -OrderDate, 2 )
firstsortedvalue ( A/B, X*Y/3 )
firstsortedvalue ( distinct PurchasedArticle, OrderDate )
firstsortedvalue ( total PurchasedArticle, OrderDate )
firstsortedvalue ( total <Grp> PurchasedArticle, OrderDate )
```

String Aggregation Functions

MinString(*[{set_expression}][**total** [<fld {, fld}>]] expression***)**

If *expression* iterated over the chart dimension(s) contains one or more values with a string representation (any text or number), the first text value in text sort order is returned, else NULL is returned.

Examples:

```
MinString( Currency )
MinString( Left( abc,2 ) )
MinString( total Currency)
MinString( <X> Currency )
```

MaxString(*[{set_expression}][**total** [<fld {, fld}>]] expression***)**

If *expression* iterated over the chart dimension(s) contains one or more values with a string representation (any text or number), the last text value in text sort order is returned, else NULL is returned.

Examples:

```
MaxString( Currency )
MaxString( Left( abc,2 ) )
MaxString( total Currency)
MaxString( total <X> Currency )
```

concat (*[{set_expression}] [**distinct**] [**total** [<fld {, fld}>]] expression[, delimiter[, sort_weight]]***)**

Returns the aggregated string concatenation of all values of *expression* iterated over the chart dimension(s). Each value may be separated by the string found in delimiter. The order of concatenation may be determined by *sort-weight*. *Sort-weight* should return a numeric value where the lowest value will render the item to be sorted first.

Examples:

```
concat( Code, ';' )
concat( FirstName&' '&LastName, ',' )
concat( distinct Code, ';' )
concat( total Name, ';' , Date )
concat( total <Grp> Name, ';' , Date)
```

Counter Aggregation Functions

count (*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression***)**

Returns the aggregated total count of values from *expression* or *field* iterated over the chart dimension(s).

For this function it is allowed to use the **distinct** qualifier in combination with the **total** qualifier This combination is not valid for any other aggregation functions.

Examples:

```
coung(Sales)
count(Price*Quantity)
count(distingct Price)
```

`count(Sales)/count(total Sales)` returns sales within the selection

NumericCount (*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression***)**

Returns the aggregated numeric count of values from *expression* or *field* iterated over the chart dimension(s).

Examples:

```
NumericCount(Sales)
NumericCount(Price*Quantity)
NumericCount(distinct Price)
NumericCount(Sales)/NumericCount(total Sales)
```

`NumericCount(Sales)/NumericCount({1} total Sales)` returns sales within the total document

TextCount (*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression***)**

Returns the aggregated text count of values from *expression* or *field* iterated over the chart dimension(s).

Examples:

```
TextCount(Sales)
TextCount(Price*Quantity)
TextCount(distinct Price)
```

`TextCount(Sales)/TextCount(total Sales)` returns sales within the selection

`TextCount(Sales)/TextCount({1} total Sales)` returns sales within the total document

NullCount (*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression***)**

Returns the aggregated count of NULL values from *expression* or *field* iterated over the chart dimension(s).

Examples:

```
NullCount(Sales)
NullCount(Price*Quantity)
NullCount(distinct Price)
```

`NullCount(Sales)/NullCount(total Sales)` returns sales within the selection

`NullCount(Sales)/NullCount({1} total Sales)` returns sales within the total document

MissingCount(*[{set_expression}][* ***distinct*** *] [* ***total*** *[<fld {, fld}>]] expression***)**

Returns the aggregated count of missing values from *expression* or *field* iterated over the chart dimension(s). Missing values are all non-numeric values.

Examples:

```
MissingCount(Sales)
MissingCount(if(Price>10, Price, 'invalid'))
MissingCount(distinct Price)
```

`MissingCount(Sales)/MissingCount(total Sales)` returns sales within the selection

`MissingCount(Sales)/MissingCount({1} total Sales)` returns sales within the total document