QlikView QlikSmart

Data Interval Ranges

Introduction

A range can be a more manageable way of slicing your data leading to more insightful analysis. When the values of a dimension (field used to categorize data) is widely spread, it can be difficult to put those values to good use – i.e. wanting to sum the sales of individuals based on their ages. When the population of this field let's say is above multiple figures, it can decrease the value of a chart that represents this – a better way to visualise this can be by creating ranges for age. This leads to less values in the field and can also use it to filter into the values that sit within each range. The examples used throughout this white paper are constructed using QlikView 12.0 SR5.

Interval Match

Interval match is a technique that allows you to match distinct numbers (or dates or time stamps) to continuous ranges (or date periods). This method is used in the script and can create distinct ranges making it extremely useful as allows the individual developing the data model more flexibility. The table that contains the distinct numbers and ranges need to be loaded before the interval match function is used, as shown below where the example used is age groups.

Start	End	AgeGroup
0	20	0-20
21	40	20-40
41	60	40-60
61	80	60-80
81		80+

Figure 1 - Age group ranges

Once this table is loaded into the data model (i.e. Table name = 'AgeRange) you will have a start and end value for each range for age group. You would then use the IntervalMatch() function on a field previously loaded in your data model which you wish to group (i.e. Age – which is a row level age field) - as part of a load statement which is used to load and match each pair of Start and End values from the Range table to every unique value of the Age field.

```
IntervalMatch(Age)
Left Join(AgeRange)
LOAD Start,
     End
Resident AgeRange;
```

The above also joins the results back into the AgeRange table – this table now has a single row for each unique value that sits in the Age field and its corresponding Start, End and AgeGroup field values. The Start and End fields can now be dropped.

```
DROP Fields Start, Stop;
```

The AgeRange table will link to table which holds the population of ages (i.e. Facts) – however we can join this to the facts table eliminating an extra link between tables. Load all values of the two remaining fields in the AgeRange table and join it to, which in this case would be your facts table along with dropping the AgeRange table. It would not create additional rows as it would be a one to one relationship.

```
Left Join(Facts)
LOAD
       Age,
       AgeGroup
Resident AgeRange;
DROP Table AgeRange;
```

Your facts table will now hold an Age field and an AgeGroup field, both of which can be used as chart dimensions/filters. Creating a drill down group can also allow you use the grouped field to drill into the more granular level.

Class() and Replace()

Using Class() is another method of creating interval ranges to categorize your data – either in the script or in chart objects. Using Class(), you specify the field name you want to create intervals for (i.e. Age) as its first parameter. The first bucket begins from the lowest value of the Age field and ends x number of values after. The value of x here would be the bucket size, which is the function's second parameter – shown below.

Class(fieldname, bucketsize) – where in this case fieldname = Age, and bucket size lets say is equal to 20. If the lowest value of the Age field was 0 and largest was 34, you would get the following as your categories.

0<=x<10	10<=x<20	20<=x<30	30<=x<40

To replace the 'x' in each class, add a third parameter with the replacement text.

To replace the '<=x<' in each class, use Replace() to wrap round Class().

```
Replace(Class(Age,20),'<=x<', '-') as AgeGroup
```

As shown here, the following parameters are used.

- First – Text field/values of which contains the text you want to replace
- Second – Text you want to replace
- Third – Text you want to replace with

TIP: The two methods detailed in this white paper can be used based on different use cases both of which are extremely useful. Interval match can be more flexible and allows you to specify exact ranges along with having the last range without an end – meaning anything greater than the start of your last value would be placed in that category. Class() is simpler to implement and can with ease create equal sized buckets before using Replace() for formatting purposses.

If you have any queries, please contact Ricky Tanna.