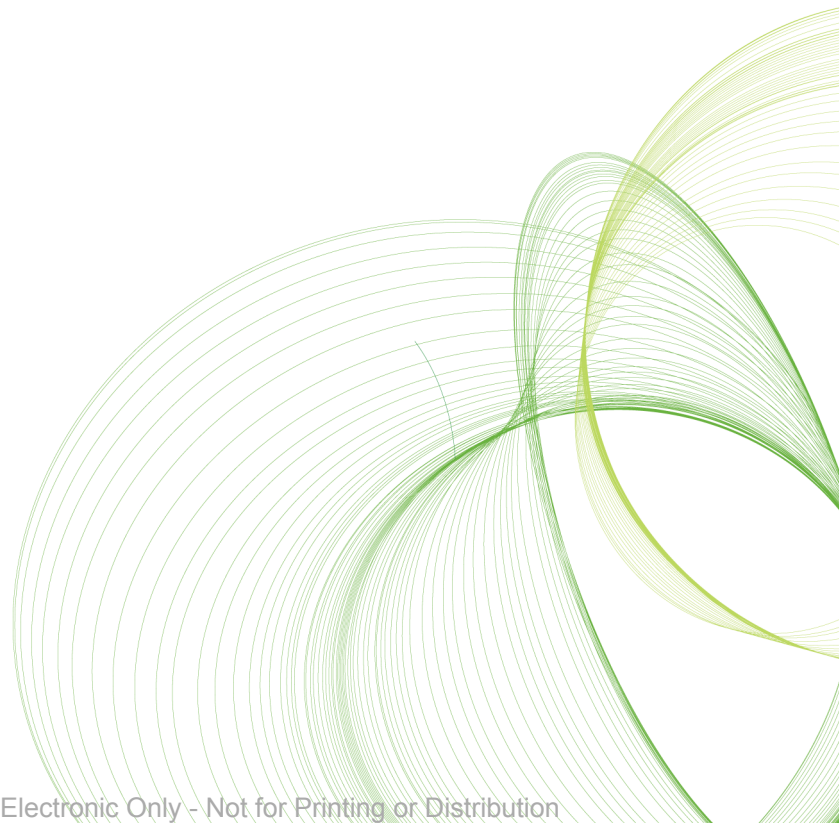
QlikView

Set Analysis

March 2010 Release

QlikView Version: 9.00 English



March 2010 Release

CONTENT

1 INTRODUCTION

Objectives

In this course:

- Set Analysis Overview
- Advanced Set Analysis
- Extending the Set Analysis Application
- Debugging the Set Analysis Application

Target Audience

This course is meant for advanced QlikView users and QlikView Developers. This is a standalone one day course focused exclusively on Set Analysis and offered as part of the standard course curriculum for QlikView Classroom Training.

This class coalesces existing material from previous QlikView classes in the Developer Series and presents new information and more advanced material.

This is the most advanced Set Analysis training available in a packaged course from QlikView.

All students in this course should have completed QlikView Developer training and have significant (six months or more) QlikView experience prior to attending.

Since students will come to this course after completing other QlikView courses, the conventions used in this manual will be familiar.

Installing the Course Materials

In some cases, this class will be taught in a virtual or hosted training environment. If you are taking the class in this setting, you can skip this section after reviewing the information on the text formatting in this manual.

In others situations, you might be asked to install the materials yourself onto a local machine. In those situations, the course materials will self-extract from the appropriate file into the default directory

C:\QlikViewTraining\SetAnalysis

Make a Windows shortcut to this folder and place it on your desktop.

Also make a Windows shortcut to the documentation folder and place it on your desktop.

C:\Program Files\QlikView\Documentation

Program versions

This course was created using the English version of QlikView 9.00 running on WindowsXP. If other operating systems or languages are used, minor differences may be noted in the visual appearance of windows and dialog boxes.

Text formats

Exercises and actions to be completed by you, the student, will be set-off with a logo, as you see, below:

Exercise/Do:

This is a sample of instructions you would see to complete an exercise containing a sequence of steps.

1 Click on the **Start** button
2 Locate the QlikView icon
3 Click on the QlikView icon to launch the program

All commands, as well as all names of menus, dialogs and buttons are in the following font style: **File - Open**

All names of list boxes, graphs and specific data in list boxes, etc. are in the following font style: *Country*

All file names are in the following font style: **QlikViewCourse.qvw**

Tips and Notes are outlined in a highlighted box, as you see below:

This sample sentence is used to illustrate important points in the text, tips and notes to consider as you complete the course materials

2 SET ANALYSIS REVIEW

Objectives

- Before Set Analysis
- Introduction to Set Analysis
- Dollar-Sign Expansion
- Identifiers
- AGGR Function
- Complete exercises using examples of each of these functions

What You Will Learn in this Chapter

This is a technical chapter for QlikView Developers. At the conclusion, you will have learned about

- What things were like before Set Analysis
- Set Analysis Basics
- Dollar-Sign Expansion, Identifiers and the AAGR function

Set Analysis

QlikView has always been good at calculating aggregates for the current selection of data. However, when you wanted to compare results for different selections in the same chart, you needed to either prepare data in the script or resort to rather complicated expressions with if clauses.

Set analysis changes all that by making it possible to modify any aggregation function with an arbitrary selection set. The set may be defined as a bookmark, as an on-the-fly selection in one or more fields, as a function of current selections, the inverse of current selections, previous selections, all data, etc.

The possibilities are endless and yet the syntax is fairly simple and straightforward.

Set Analysis Defined

Set Analysis is an alternative set of records and can be defined by a set expression. Hence, a set is conceptually similar to a selection.

A set expression always begins and ends with curly brackets when used.

```
=Sum({$<Year += {2008, 2009} >}LineSalesAmount)
```

Figure 1. A set expression

Before Set Analysis

Prior to the availability of Set Analysis, there was no way to access non-selected, non-possible data in a calculation without pre-calculating data in the load script or using the **ALL** qualifier.

A Set Analysis specification can be included in any expression calculation, is always evaluated over all data, and is not itself limited by a chart dimension. A chart expression containing a Set Analysis specification, however, must still ultimately be limited by the dimensionality of the cell being calculated.

A schematic picture of the data in QlikView could, for example, be represented in the figure, below. We have a number of values (here represented by a bunch of dots) with different characteristics. In this case they have different color. We can select all the blue dots (for example, to count them).

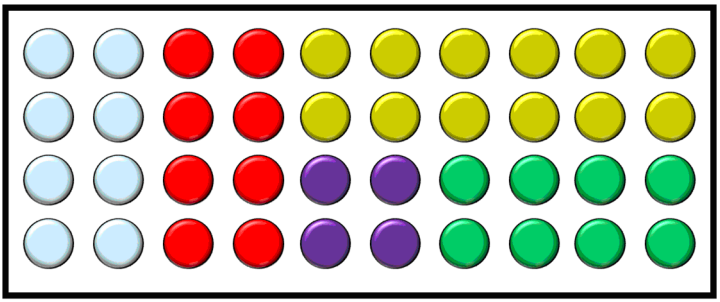


Figure 2. A schematic picture of a Data Set in QlikView

In QlikView, the selection state becomes green and we can easily see that there are eight blue dots (figure, below). Everything else disappears, so to speak from QlikView's consciousness. It is gray and no longer visible in our summation.

What if we are suddenly interested to know how many yellow dots we had in our image? You might just remember, and, of course, we can still clear our selection and instead choose the yellow. The challenge is to show them side by side.

Figure 3. One selected part of the Data Set

Using Set Analysis

If we use set analysis and make the same selection as before. The difference is that the rest of the piece now does not disappear completely, but it is still available for further calculations. We can (actually in several different ways) select a sample of the yellow dots and compare with the light blue. We can even determine the data that is outside the two samples (!) And compare with sample 1 and 2.

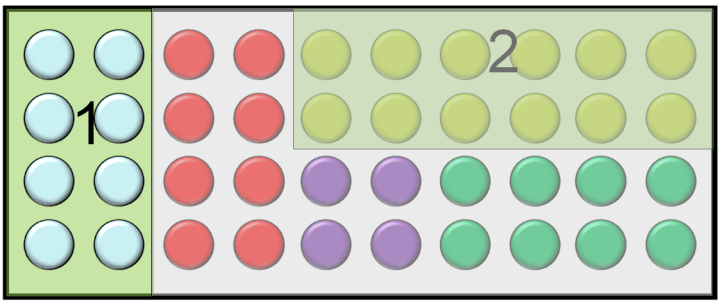


Figure 4. Comparing different parts of the data set with Set Analysis

Overview

A selected value

In this example we will start to create a situation where it becomes easier to manage the previous year's sales without having to change the load script. It is common to use the date flags with the corresponding report options. At least in theory we would now be able to throw out the solution completely. This expression shows how we could enter the last year sales compared to currently selected year when we use the function **Only**().

Sum({**$<Year={$(=only**(Year)-1)}>}Sales)

Comparison between two selections

We can also compare two completely different selections. The easiest way (there are several) is to create a bookmark for each selection you want to compare and then to use this feature

Sum({BM01} Sales) **vs. Sum**({BM02} Sales)

What is NOT selected

When you make a selection, by definition, there is going to be data that is not selected. Before Set Analysis we used **TOTAL** and **ALL** functions, but these workarounds were limited, and we might still not have achieved everything we wanted. Using Set Analysis, we can show what is **not** chosen.

Sum({1-$} Sales)

Always a selected value

We could also handle the "always a selected value" in a new way. As every experienced QlikView developer will know, it is possible to set a listbox so that a value is always chosen. Now we can achieve almost the same with Set Analysis, as below:

Sum({**$<Currency**=*{$(=firstsortedvalue(Currency))}>}* **Sales**)

Aggregated functions

Sets can also be used in aggregation functions. Aggregation functions normally aggregate over the set of possible records defined by the current selection. But an alternative set of records can be defined by a set expression. Hence, in this case, a set is conceptually similar to a selection.

Basic Components in Set Analysis

The purpose of Set Analysis is to let an expression use a selection other than the one made in the layout. A Set Analysis expression is built out of three components, the identifier, the operators and a modifier.

Figure 5. Identifying the three different components in a Set Analysis expression: identifier, operators and modifier.

Basic aggregation to sum sales.

=Sum(*LineSalesAmount*)

Set analysis, basic syntax {$< >} starts and ends with curly brackets.

$=Sum({$< >} *LineSalesAmount*)

The $-dollar sign is defined as an identifier of the data set.

Start and end of Set Modifier Clause

<…….>

Year += {2008, 2009}

and between the <> we find the modifier that will define the selection. The modifier can contain selections in different fields, hard coded values, functions(), variables

+=

defined as an operator, in this example it makes the current selection of year a union with Year 2008 and 2009.

Basic components - Set Identifiers

There is a constant that can be used to denote a record set; **1**. It represents the full set of all the records in the application.

The **$** sign represents the records of the current selection. The set expression {$} is thus the equivalent of not stating a set expression.

{1-$} defines the inverse of the current selection, i.e. everything that the current selections excludes.

Selections from the Back/Forward stack can be used as set identifiers, by use of the dollar symbol: **$1** represents the previous selection, i.e. equivalent to pressing the Back button.

Similarly, **$_1** represents one step forward, i.e. equivalent to pressing the Forward button. Any unsigned integer can be used in the Back and Forward notations, i.e. **$0** represents the current selection.

Finally, bookmarks can be used as set identifiers. Note that only server and document bookmarks can be used as set identifiers. Either the bookmark ID or the bookmark name can be used, e.g. **BM01** or **MyBookmark**. Only the selection part of a bookmark is used.

Set Identifiers: Examples

Once again, a Set Identifier is a constant that can be used to denote a record set. Some examples follow.

Sum({1} LineSalesAmount) =

A full set of all the records in the application, disregarding the selection but not the dimension.

Sum({1} Total LineSalesAmount) =

A full set of all the records in the application, disregarding the selection both selection and dimension.

Sum({$}LineSalesAmount) =

The records of the current selection.

Sum({1-$} LineSalesAmount) =

The inverse of the current selection

Sum({$1} LineSalesAmount) =

Previous selection (equivalent to press Back)

Sum({$_1} LineSalesAmount) =

Next selection (equivalent to press Forward). Only relevant if you just made one Back operation.

Sum({BM01}} LineSalesAmount) or Sum({MyBookmark}} LineSalesAmount) =

Only the selection part of a bookmark is used. Variable values are not included. Thus, it is not possible to use input fields in bookmarks for set analysis.

Sum({Server\BM01} LineSalesAmount) =

For the server bookmark BM01.

Check the Different Identifiers Example in the Application

In the following exercise you will get a chance to evaluate the different identifiers.

Change vSet_Identifier {1-$} or Change vSet_Operator

Salesperson	Sum (LineSalesAmount)	Sum({1} LineSalesAmount)	Sum({$} LineSalesAmount)	Identifier {1-$}
	106,353	**13,321,238**	**106,353**	**4,512,642**
Brolin, Helen	0	1,959,870	0	1,959,870
Callins, Joan	42,698	732,899	42,698	0
Carsson, Rob	1,706	2,625,611	1,706	0
Hendrix, Ingrid	20,007	637,644	20,007	0
Lindwall, Tom	19,560	2,070,182	19,560	0
Presley, Erik	0	509,456	0	509,456
Roll, Frank	0	2,043,316	0	2,043,316
Shine, Leif	9,257	1,902,031	9,257	0
Skoglund, Lennart	13,125	840,228	13,125	0

Figure 6. In Exercise A (in the separate chapter at the end of the book), evaluate different Identifiers in the existing chart in the QlikView training file, Exercise A.

Basic components - Set Operators

Important: Set Analysis opens up the opportunity to access all loaded data in a QlikView document at any time, regardless of selection state.

Note: A set expression is always enclosed in curly brackets when used, e.g.**{**BM01**}**.

Several operators are used in set expressions. All set operators use sets as operands, as described above, and return a set as result. The operators are as follows:

+ Union. This binary operation returns a set consisting of the records that belong to any of the two set operands.

– Exclusion. This binary operation returns a set of the records that belong to the first but not the other of the two set operands. Also, when used as a unary operator, it returns the complement set.

*** Intersection**. This binary operation returns a set consisting of the records that belong to both of the two set operands.

/ Symmetric difference (XOR). This binary operation returns a set consisting of the records that belong to either, but not both of the two set operands.

The order of precedence is

1 Unary minus (complement)
2 Intersection and Symmetric difference
3 Union and Exclusion.

Within a group, the expression is evaluated left to right. Alternative orders can be defined by standard brackets, which may be necessary since the set operators do not commute, i.e. A + (B – C) is different from (A + B) – C which in turn is different from (A – C) + B.

Operators are used as operands to combine different set, and return a set as result

+ Union =A + B+C

- Exclusion =A

***** Intersection =B

/ Symmetric difference (XOR): = A + C

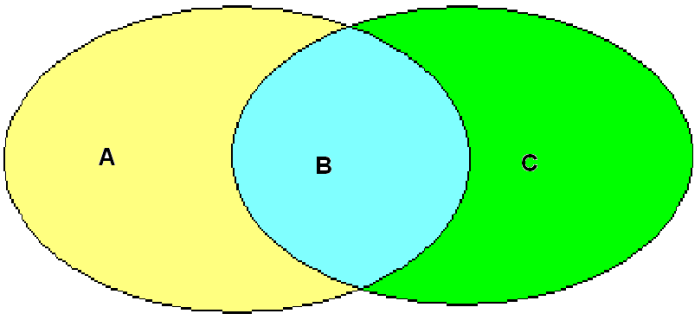


Figure 7. An Operator combines different data sets

Set Operator Examples:

sum({1-$} LineSalesAmont **)**

returns the sales for everything excluded by the current selection.

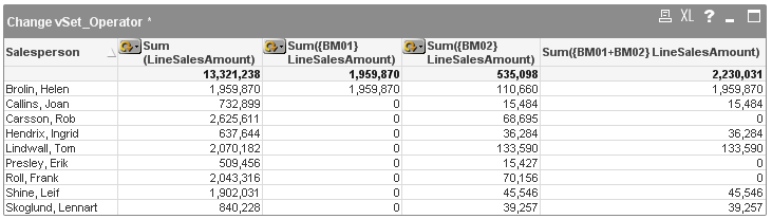
sum({$*BM01} LineSalesAmont**)**

returns the sales for the intersection between the current selection and bookmark BM01.

sum({-($+BM01)} LineSalesAmont**)**

returns the sales excluded by current selection and bookmark BM01.

Note: The use of set operators in combination with basic aggregation expressions involving fields from multiple QlikView tables may cause unpredictable results and should be avoided. For example, if *Quantity* and *Price* are fields from different tables, then the expression **sum**({$*BM01} *Quantity * Price*) should be avoided.



Change vSet_Operator *

Salesperson	Sum (LineSalesAmount)	Sum({BM01} LineSalesAmount)	Sum({BM02} LineSalesAmount)	Sum({BM01+BM02} LineSalesAmount)
	13,321,238	**1,959,870**	**535,098**	**2,230,031**
Brolin, Helen	1,959,870	1,959,870	110,660	1,959,870
Callins, Joan	732,899	0	15,484	15,484
Carsson, Rob	2,625,611	0	68,695	0
Hendrix, Ingrid	637,644	0	36,284	36,284
Lindwall, Tom	2,070,182	0	133,590	133,590
Presley, Erik	509,456	0	15,427	0
Roll, Frank	2,043,316	0	70,156	0
Shine, Leif	1,902,031	0	45,546	45,546
Skoglund, Lennart	840,228	0	39,257	39,257

Figure 8. In Exercise B (in the separate chapter at the end of the book), evaluate the result of using different Operators in the existing chart in the QlikView training file, Exercise B.

Basic Components - Set Modifiers

A set can be modified by making an additional or a changed selection.

Such a modification can be written in the set expression. Also implicit intersections, exclusions and symmetric differences can be defined using "*=", "–=" and "/=".

Finally, for fields in and-mode, there is also the possibility of forced exclusion.

If you want to force exclusion of specific field values, you will need to use "~" in front of the field name.

Examples of expressions using set analysis

=Sum({$ **< Year={2007} >**} Sales)

Same as select year = 2007

=Sum({$ **< Year={">=2007"} >** } Sales)

Same as a text search in Year *>= 2007*

=Sum {$ **<Year = {$(=max(Year))} >** } Sales)

Use of a function to select the set of Years using an expression

Examples of Set Modifiers

A set can be modified by making an additional or a changed selection.

<Field={2007,2008}>
Select only the records where the field Field = 2007 or 2008

<Field=>
Ignore selections made in the field Field

<Field={"*"}>
Select all in the field Field

<Field={}>
Select records that are not associated with the selection in the field *Field*

<Field=Field + {2007,2008}> or <Field+={2007,2008}>
Current selection in the field Field + Field = 2007 or 2008

Syntax example: Dissection of a set analysis expression

=Sum({$ < Year={2007, 2008}, Month= >} LineSalesAmount)

=Sum(LineSalesAmount)

Basic aggregation to sum sales

{}

Start and end of set analysis statement

$

Identifier. In this case $ = Current Selection

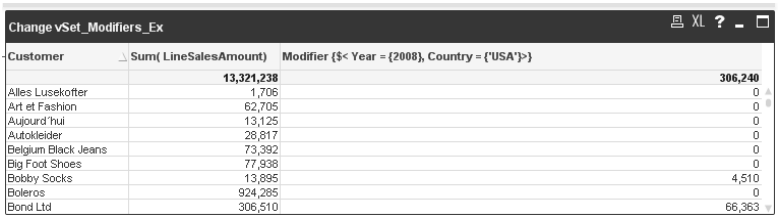
<.....>

Start and end of Set Modifier Clause

Year={2007, 2008}

Set of Modifier Element, values are separated by comma, and when there is more than one Modifier Element, they are separated by comma.

< Year={2007, 2008}, Month= >


Change vSet_Modifiers_Ex

Customer	Sum(LineSalesAmount)	Modifier {$< Year = {2008}, Country = {'USA'}>}
	13,321,238	306,240
Alles Lusekofter	1,706	0
Art et Fashion	62,705	0
Aujourd'hui	13,125	0
Autokleider	28,817	0
Belgium Black Jeans	73,392	0
Big Foot Shoes	77,938	0
Bobby Socks	13,895	4,510
Boleros	924,285	0
Bond Ltd	306,510	66,363

Figure 9. In Exercise C (in the separate chapter at the end of the book), Evaluate examples of different Modifiers in the existing chart in the QlikView training file. Exercise C.

Syntax for Set Analysis

The full syntax (not including the optional use of standard brackets to define precedence) is

set_expression ::= { set_entity { set_operator set_entity } }

set_entity ::= set_identifier [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 ADVANCED SET ANALYSIS

Objectives

- Advanced Set Analysis Functions
- Modifiers with searches, wildcards
- $-sign expansion using functions and variables
- **P** and **E** Element functions
- Aggregated functions
- Data islands
- **Concat**() function

What You Will Learn in this Chapter

When using Set Analysis, you often want to compare the set chosen to a selection made in the layout. You need to define a set to compare with. Sometimes this set is dependent on your selection. This means that you do not want to have predefined values in the set of your expression but rather a value calculated from the selection you have chosen. In cases such as this, you need to use dynamic modifiers such as variables.

In this chapter we will look closely at some of the more advanced functionality that can be used in Set Analysis.

Set Modifiers with Explicit Field Value Definitions and searches

The modifier consists of one or several field names, each followed by a selection that should be made on the field, all enclosed by < and > as in

<Year={2007, 2008}, Region={US}>

Field names and field values can be quoted as usual, e.g.

<[Sales Region]={'West coast', 'South America'}>.

There are several ways to define the selection.

A simple case is a selection based on the selected values of another field, e.g. <OrderDate = DeliveryDate>. This modifier will take the selected values from "*DeliveryDate*" and apply those as a selection on "*OrderDate*".

Note: If there are many distinct values – more than a couple of hundred – avoid this operation because it is CPU intensive.

The most common case, however, is a selection based on a field value list enclosed in curly brackets, the values separated by commas, e.g. <Year =

{2007, 2008}>. The curly brackets here define an element set, where the elements can be either field values or searches of field values.

A search is always defined by the use of double quotes, e.g.

<CategoryName = {"*Clothes*"}>

will select all ingredients including the string 'clothes' by using wildcard stars *.

Note: Searches are case-insensitive and are made over excluded values too.

Further, the selection within a field can be defined using set operators and several element sets, such as with modifier <Year = {"20*", 1997} - {2000}> which will select all years beginning with "20" in addition to "1997", except for "2000".

During the exercises we will learn how to use set analysis in different ways to compare data over time.

Tip: Empty element sets, either explicitly e.g. < Year = {}> or implicitly e.g. <Year = {1872}> (a search which returns no values) will result in a set of records that are not associated with any product.

Exercise 1

Note: in this edition of the Set Analysis course, all of the exercises have been placed in a separate chapter at the end of the book.

Set Modifiers with Explicit Field Values, Comparing Time Dimensions in Straight table.

Prior to the availability of Set Analysis, there was no way to access nonselected/non-possible data in a calculation without pre-calculating data in the load script or using the ALL qualifier.

The Set Identifier **{1} Total** is equal to a full set of all the records in the application, disregarding the selection both selection and dimension.

A Set Analysis specification can be included in any expression calculation, is always evaluated over all data, and is not itself limited by a chart dimension.

A chart expression containing a Set Analysis specification, however, must still ultimately be limited by the dimensionality of the cell being calculated.

Exercise 1 a, b Extra

Set Modifiers with Explicit Field Values and the Set Identifier [1} Total, Set Expression in Gauge chart, Text box

Set Modifiers with $-sign expansions

To make the Set Modifier flexible and avoid maintenance of an expression and application a $-sign expansion is often to prefer instead of writing explicit values. By using functions or variables you create a dynamic set expression.

Exercise 2 Extra

Set Modifiers with $-sign expansions, Comparing Time Dimensions using function **Only()** in a straight table.

Exercise 3

Set Modifiers with $-sign expansions, Comparing Time Dimensions using function **Max()** in a straight table.

Measures displayed in charts and graphs normally show the aggregated set of possible records defined by the current selection. Set Analysis allows you to override those selections and to display alternate sets of records.

Set Analysis is a departure from the standard QlikView associative model. Though this model works well for most situations, it is many times necessary to select one or multiple alternate sets for comparison. As an example, if you want to select a *Year* and compare it to the *Previous Year*, the standard associative model will exclude the *Previous Year*.

Alternate sets created through Set Analysis can be static or dynamic. How you design Set Analysis statements is critical to providing the correct alternate set for comparison. If you overuse Set Analysis or use it incorrectly, you run the risk of making the application much like a cube in that the much of the data displayed is in pre-defined sets.

The above notation defines new selections, disregarding the current selection in the field. However, if you want to base your selection on the current selection in the field and add field values, e.g. you may want a modifier

<Year = Year + {2007, 2008}>.

A short and equivalent way to write this is

<Year += {2007, 2008}>

i.e. the assignment operator implicitly defines a union.

Remember: Set Analysis provides the opportunity to access all loaded data in a QlikView document at any time, regardless of the current selection state.

Finally, for fields in and-mode, there is also the possibility of forced exclusion.

If you want to force exclusion of specific field values, you will need touse "~" in front of the field name.

Exercise 4

Set Modifier with $-sign expansion, Comparing Time Dimensions using variables for functions in a straight table.

Exercise 5

Set Modifier with a Searcher using wildcards and implicit set operator +=

Salesperson += *Presley*

Set Modifiers with Aggregation Functions

Sets can be used in aggregation functions. Aggregation functions normally aggregate over the set of possible records defined by the current selection. But an alternative set of records can be defined by a set expression. Hence, a set is conceptually similar to a selection. Several operators are used in set expressions. All set operators use sets as operands, as described above, and return a set as result.

- The expressions can be complex and difficult to follow
- Set Analysis represents a departure from standard QlikView functionality

Recommendation: Make sure that you clearly describe the data displayed by an alternate set. Otherwise, users may not understand why their selections have no effect or an unexpected effect on the data displayed.

Set Modifiers with Implicit Field Value Definitions

In the Set Analysis material you have learned so far, all of the field values in a Set Analysis statements have been explicitly defined or defined through searches.

There are, however, additional ways to define a set of field values by the use of a nested set definition. The nested values can use Element Functions or Aggregation Functions.

Set Modifiers with Element Functions P() and E().

One method of implicitly defining a set is through the use of the element functions **P**() and **E**(), representing the element set of possible values and the excluded values of a field, respectively.

Set Modifiers Implicit Field Value Definition

Using P() and E() element functions

In the standard Set Analysis course material, you learned the basics of creating Set Analysis statements. There are several other functions that can be very useful in defining alternate sets.

Example:

The following statement represents the Sum of all Sales for the set of Customers who purchased Product 'A':

Sum({$<Customer = (P({1<Product = {'A'} >})) >}Sales)

The element function **P**( ) here returns a list of possible customers; those that are implied by the selection 'A' in the field Product.

(P({1<Product = {'A'} >}))

The next example represents the Sum of all Sales for the set of Customers who did NOT purchase Product 'D':

sum({$<Customer = (E({1<Product = {'D'}>})) >}Sales)

Suppose Product A is a major appliance and Product D is a maintenance plan. If you want a list of all Sales for Customers who bought the appliance but not the plan, you could combine these using the Intersection Set Operator (*) and end up with the following:

Sum({$<Customer = (P({1<Product = {'A'} >})) * (E({1<Product = {'D'}>})) >}Sales)

Exercise 6

Set Modifiers with Implicit Field Value Definitions using Element Functions **P**()

Exercise 7

Set Modifier Advanced Searcher using aggregation with **P** Function

Data islands and CONCAT function

What is a Data Island?

There are sometimes cases where you may want to create a tie between values in two tables where no link exists. Tables that are not linked to any other tables in the QlikView Dataset are called Data Islands. The **CONCAT** function provides a method to do this.

CONCAT 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.

Set Analysis using concat() and Data Island

In the next exercise we will use the **CONCAT**() function to build up a list from an data island field.

Creating a Data/Date Island

In the sample application there are some Data Islands already created in the script.

It is always a good idea to load data islands distinctly to reduce data load.

```
Qualify *;
DataIsland_1:
Load distinct
SalespersonName
resident Salesperson;

DataIsland_2:
Load distinct
CategoryName
resident Products;

DataIsland_3:
Load distinct
SupplierName
resident Products;

Unqualify *;
```

Those script lines will create fields with no link to any other data sets in the QlikView Data Model in this application.

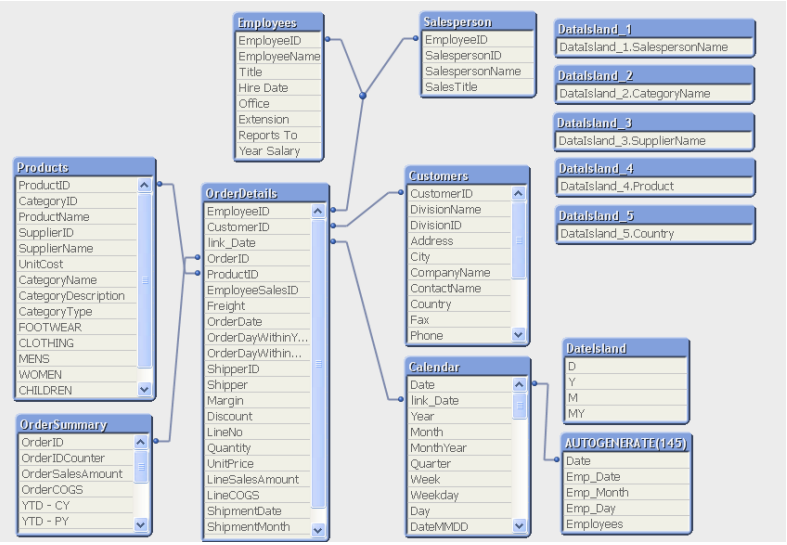


Figure 1. None of the Data Islands has a link to any other data sets in the QlikView Data Model.

Exercise 8
Set Modifier **CONCAT**() function and **MONTHEND**() function

Exercise 9
Set Modifier **CONCAT**() function and Data Island

Data Island DataIsland_1.SalespersonName

Exercise 10
Set Modifier **CONCAT**() function, Data Island and variable *vSalesperson-Name* Data Island and Variable *vSalesperson*

Advanced Set Analysis Aggregation Functions

A second method of implicitly defining a set is through the use of Aggregation Functions, representing the set of possible values or the excluded values of a field based on the results of an aggregation.

Example:
The following statement represents the Sum of all Sales for the set of Customers who purchased Product 'A':

sum({$<Customer={"=sum({1<Product = {'A'}>}Sales)>0"}>}Sales)

The aggregation function **sum**() here returns a list of possible customers; those that are implied by Sales of Product A greater than 0.

{"=sum({1<Product = {'A'}>}Sales)>0"}

The next example represents the Sum of all Sales for the set of Customers who purchased Product 'D':

sum({$<Customer={"=sum({1<Product = {'D'}>}Sales)>0"}>}Sales)

The aggregation function **sum**() here returns a list of possible customers; those that are implied by Sales of Product D greater than 0.

Again, as in the Element Function example, we are trying to build the set of all Customers who purchased Product A but did not purchase Product D. You could combine these using the Exclusion Set Operator (-) and end up with the following:

sum({$<Customer={"=sum({1<Product = {'A'}>}Sales)>0"} -
{"=sum({1<Product = {'D'}>}Sales)>0"}>}Sales)

Note that the Exclusion Set Operator excludes rows defined in the first half of the expression, in other words, first, select all Customers who purchased A, including those who purchased D, and then exclude those who purchased D from that set.

Exercise 13
Set Modifier Advanced Search using aggregation and variables

Advanced Set Analysis Functions

Exercise 11 Extra
Set Modifier using **CONCAT**(), DataIsland, *vCategoryName*, *vSupplierName*

Exercise 12
Set Modifier using **CONCAT**, DataIsland, variables and **P** and **E** function From Exercise 6.

4 Extending Set Analysis

Objectives

- Dynamic Set Analysis
- Indirect Set Analysis
- Best practice

Extending Set Analysis Highlights

- Why and how to use Dynamic Set Analysis
- Why and how to use Indirect Set Analysis
- Step by step instructor show and tell followed by exercises to extending the capabilities of the application
- Explain best practices
- Basic troubleshooting your extended application (transition to the next chapter)
- Step by step instructor show and tell followed by exercises to extending the capabilities of the application
- Explain best practices
- Using Dynamic Set Analysis,

Exercise 14 Extra

Note: in this edition of the Set Analysis course, all of the exercises have been placed in a separate chapter at the end of the book.

Check SA Time Report 1 Sheet in Solution application for Set Analysis Course.qvw.

This sheet contains two reports with set expressions to compare different time periods. Can be used to copy and paste. Apply into other applications with small modifications.

Exercise 15 Extra

Check SA Time Report 2 Sheet in Solution application for Set Analysis Course.qvw.

This sheet shows two different ways of how to use Dynamic Set Analysis for comparing different periods in reports.

The first by using variables for time functions and the second by creating variables for a complete set expression.

5 Error Checking Set Expressions

Objectives

- Recognize problems with resulting data
- Define strategies for debugging
- Understand and fix errors
- Validate resolution

What You Will Learn in this Chapter

Based on the work on exercises from the previous chapter(s)

- Recognize that there is a problem with the resulting data
- Define strategies for working through the debugging process with Set Analysis
- Understand the error and how to fix it
- Fix it
- Show that the resolution works

Create, modify and evaluate your set expression by writing the expression in a input box.

One input box and a straight table showing the result of the set expression can be useful to add to the application during the development of Set expressions. Using a cyclic group of the most common dimensions in the application as a dimension in the table makes it easy to evaluate whether or not your Set expression presents the desired results.

The input box makes it easier to make changes in the expression and still see the result in the straight table.

A helpful QlikView-feature is of course, to look at the color coded syntax help, in the expression editor. When you use an input box you have to add an = (equal-sign) before the expression to see the color coded syntax help.

Basic Troubleshooting for Your Extended Application

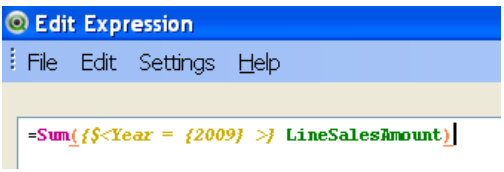


Figure 1. Checking the Set Analysis expression

Check the result of the set analysis expression in the chart to confirm the right result is achieved. If not, check the syntax is correct in the expression editor or/and make sure the used function or variable separately gives the desired result.

When there is an error in expression you will find this message in the expression editor at the left hand upper corner. Check correct spelling of field-names (case sensitive), the brackets to always correspond with another left or right, as well as the curly brackets, all field values must be enclosed by curly brackets. Functions and variables must start with a $-sign and enclosed by brackets.

In the example below the red left curly bracket indicates an error, make sure there is a corresponding right curly bracket.

When there are functions and variables in the set expression, test the expression with an explicit value to know what format is needed, especially with date and text fields, which can be tricky.

Functions and variables can be tested separately in a text box.

Take a close look at the expression and try to find out what's wrong.

*Figure 2. The curly bracket after "*Clothes*" is missing.*

Once corrected, the expression should look like this:

Exercise 16

Note: in this edition of the Set Analysis course, all of the exercises have been placed in a separate chapter at the end of the book.

Evaluate your Set Expression in the Basic Component Sheet

Exercise 17

Create an advanced set analysis expression step by step.

6 SUMMARY AND CONCLUSIONS

Objectives

- Review course objectives
- Summarize course conclusions

What You Will Learn in this Chapter

- Review course objectives
- Summarize conclusions

Review and Summary

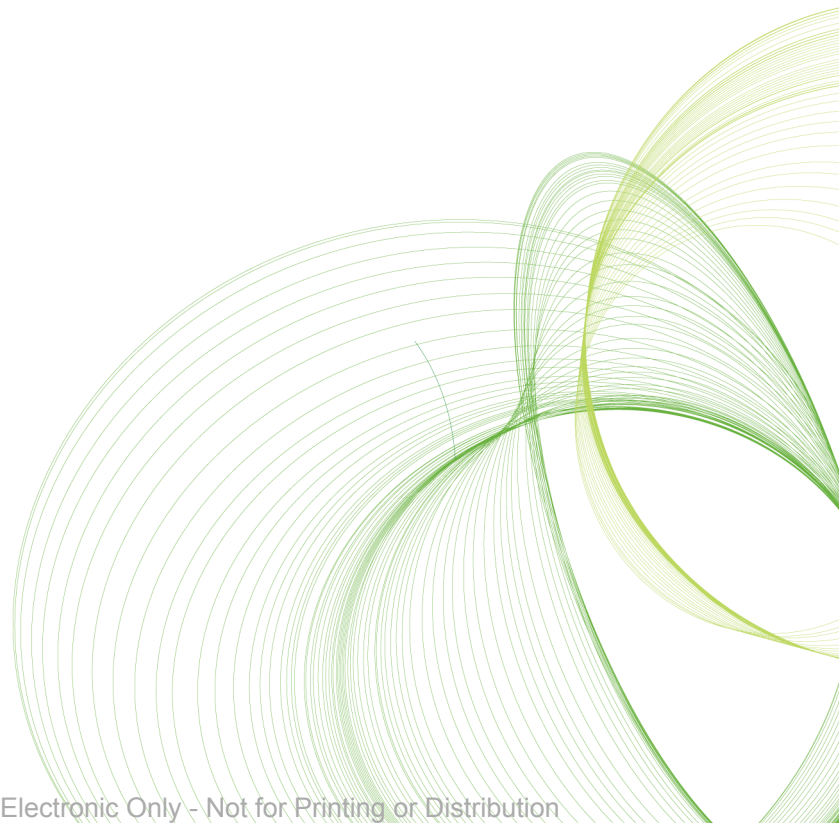
- Where to go from here, other resources, enablement sessions, virtual training resources, seminars and demonstrations
- Set Analysis is an excellent new way of working with QlikView and it opens up a wide range of possibilities that was impossible or difficult to achieve before Set Analysis. It makes it easy to compare different Sets of selections, sometimes disregarding the actual selections in the layout.
- Although it is a new way of working with QlikView and opens up a wide range of functionality, a developer must handle Set Analysis carefully. A user often find it hard to look at Set Analysis expressions if they are not clearly stated as to what is actually shown.
- A developer must also be careful and make sure not to use Set analysis in ways that slow the application down. Set analysis can be CPU intense and thus Set Analysis should be tested in work situations to make sure that it does not slow the application down.
- Since we tend to read more and more data into QlikView, we need to consider if we can create the functionality of a Set Analysis expression directly in the QlikView script. A good guideline is that if it is possible to do it in the script, then do it in the script. The load time of a script can often be reduced in different ways and hence as much work as possible should be placed in the load script instead of in the user interface objects of QlikView.

QlikView

Set Analysis Exercises

March 2010 Release

QlikView Version: 9.00 English



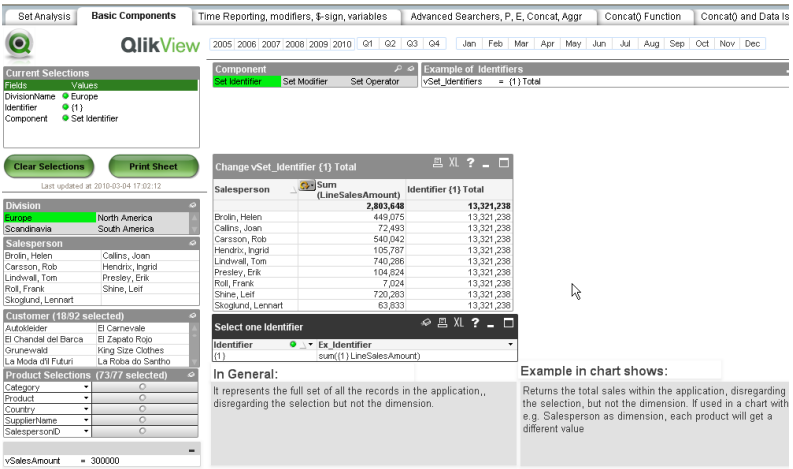
March 2010 Exercises Release

Exercises

This section of the course manual contains all of the exercises used in the class, as well as some extra credit examples.

Exercise A: Evaluate the Set Analysis Identifier

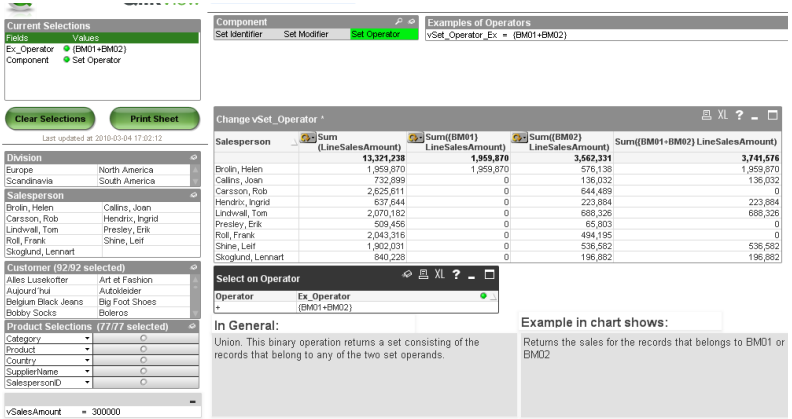
Do:



1 Click on the **Start** button
2 Locate the QlikView icon
3 Click on the QlikView icon to launch the program **File|Open** and choose the training file (**Set Analysis Course.qvw**)
4 Navigate to the **Basic Component Sheet**.
5 Select **Set Identifier** in the list box **Component**
6 Select one of the **Set Identifier** you would like to explore from the input box
7 Check the result in the third column of the Straight Table
8 Compare the result with the first expression column
9 Make selections in the list boxes and see how the result changes
10 Select one row from the table box on the bottom of the screen and see the description of the identifier.
11 Leave the application open for the next exercise.

Exercise B: Evaluate the Set Analysis Operator

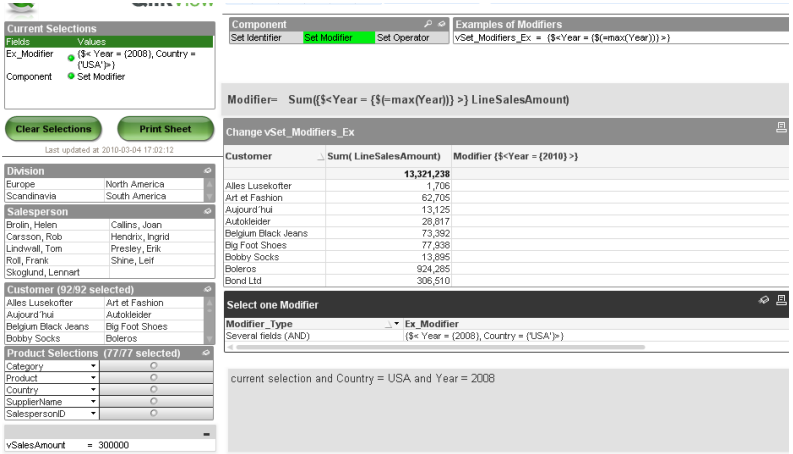
Do:



1 Navigate to the **Basic Component Sheet**.
2 Select **Set Operator** in the list box **Component**
3 Select one of the Operator you would like to explore from the input box
4 Check the result in the fourth column of the Straight table
5 Compare the result with the other expression columns
6 Make selections in the list boxes and see how the result changes
7 Select one row from the table box on the bottom of the screen and see the description of the operator.
8 Leave the application open for the next exercise.

Exercise C: Evaluate the Set Analysis Modifier

Do:



1 Navigate to the **Basic Component Sheet**.
2 Select **Set Modifier** in the list box **Component**
3 Select one of the **Modifier** you would like to explore from the input box
4 Check the result in the forth column of the Straight table
5 Compare the result with the other expression columns
6 Make selections in the list boxes and see how the result changes
7 Select one row from the table box on the bottom of the screen and see the description of the Modifier.
8 Leave the application open for the next exercise.

Advanced Set Analysis Exercise 1

Do: Modifier with Explicit Field Value Definitions

Create a straight table showing sales in an Annual Comparison between year 2007 and 2008, for all Divisions using a Set Analysis Modifier. Show for all Divisions using a Set Analysis Modifier. Show the sales for each year and the differences in amount and percentage. The first expression will show the Sales for the current selections.
Try to find out advantages and disadvantages with using an explicit field value.

```
Modifier=  Sum({$<Year = {2008} >} LineSalesAmount)
```

Set Modifier Hard Coded Year 2008 vs 2007, ex 1

Division	sum (LineSalesAmount)	2008	2007	2008 vs 2007	2008 - 2007%
	13,321,238	4,891,125	3,058,936	**1,832,189**	**37.46%**
Europe	2,803,648	1,040,585	537,014	**503,571**	**48.39%**
North America	2,887,815	1,073,055	689,399	**383,656**	**35.75%**
Scandinavia	3,281,675	1,189,136	883,073	**306,062**	**25.74%**
South America	982,443	345,926	252,313	**93,613**	**27.06%**
X	3,365,657	1,242,423	697,137	**545,286**	**43.89%**

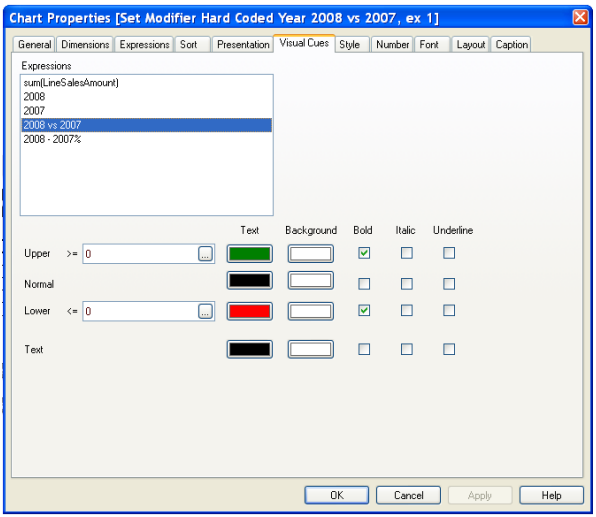
Do:

1 Navigate to the **Time Reporting, Modifiers, $-sign, Variables** Sheet.
2 Right Click - **New Sheet Object | Chart | Straight Table**
3 On the **General Tab**, **Name Window type: "Set Modifier Hard Coded Year 2007 vs 2008, ex 1"**
4 On the **Dimensions Tab**, add dimension: **DivisionName**
5 Change the dimension label to "**Division**".
6 On the **Expressions Tab**, create the following five **Expressions** using the **Labels** provided:
Label 1 -
=Current Selection
Expression 1-
sum(LineSalesAmount)
Label 2-
=2008
Expression 2-
= sum(*{$<Year = {2008} >}* LineSalesAmount)
Label 3-

= 2007
Expression 3-
= sum({$<*Year* = *{2007}* >} LineSalesAmount)
Label 4-
=2008 vs 2007
Expression -4
=Column(2)-Column(3)
Label *5*-
=2008 vs 2007
Expression *5*
=(Column(2)-Column(3))/ Column(2)

7 Click **Finish**

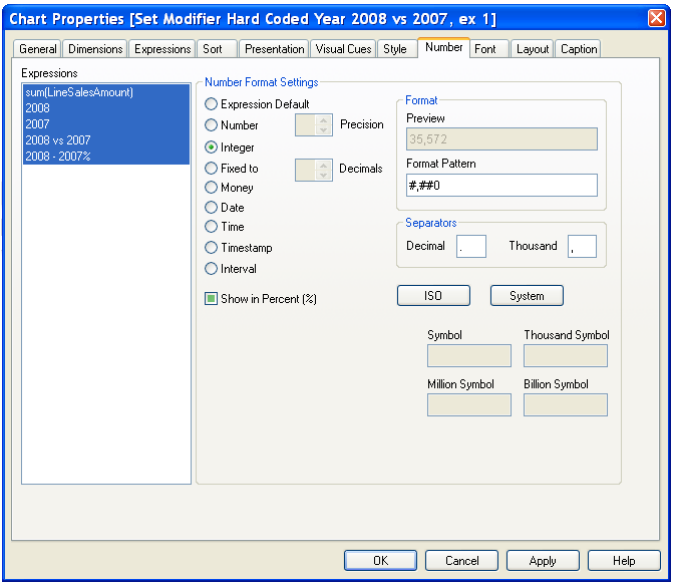
8 On the **Visual Cues** tab, make the negative values for the year-to-year comparison red and the positive values green.



9 **Save** your QlikView file and then continue to edit the**: "Set Modifier Hard Coded Year 2007 vs 2008, ex 1"** straight table.

10 Set the **Sort order** to match the depiction, above, remembering that **Division** should be set to **Text**.

11 Continue to the Number tab and set all expressions to Integer.



12 Click **OK** to save the table.
13 **Save** the application
14 Leave the application open for the next exercise.

Advanced Set Analysis Exercise 1a

Modifier with Explicit Field Value Definitions

In this exercise we will change the expression in the existing Gauge chart to show the relative Sales for Europe compared to all other Sales. To select Europe in the expression we use a set modifier and to compare it to all sales in the application for that purpose we use the identifier 1 equal to all records.

```
Expression =  Sum({$< DivisionName = {"Europe"}>} LineSalesAmount)
```

```
Expression =  Sum({1} Total LineSalesAmount)
```

Do:

1 Navigate to the **Time Reporting, Modifiers, $-sign, Variables** Sheet.
2 Open the properties for the **Gauge Chart**.
3 On the expression tab change the expression to:
Sum*({$< DivisionName = {"Europe"}>}* **LineSalesAmount) /Sum***({1} Total LineSalesAmount*)
4 Click **Finish**
5 Click **OK** to save the chart.
6 **Save** the application
7 Leave the application open for the next exercise

Advanced Set Analysis Extra Exercise 1b

Modifier with Explicit Field Value Definitions

In this exercise we will change the expression in the existing Text box over the Gauge chart to show the relative Sales for Europe compared to all other Sales. Use the same expression as in the Gauge chart and add explanatory text and format the percentage number.

Sales in % of total Sales = +16.0%

```
Expression =  Sum({$< DivisionName = {"Europe"}>} LineSalesAmount)
```

```
Expression =  Sum({1} Total LineSalesAmount)
```

Do:

1 Navigate to the **Time Reporting, Modifiers, $-sign, Variables** Sheet.
2 Open the properties for the existing text box above the **Gauge chart**.
3 On the **General tab** change the expression to:

```
='Europe = ' & num(Sum({$< DivisionName = {"Europe"}>} LineSalesAmount)
/
Sum({1} LineSalesAmount), '+#,###.#%;-#,###.#%')
```

4 Click **OK**
5 **Save** the text box
6 **Save** the application
7 Leave the application open for the next exercise

Advanced Set Analysis Exercise 2

Dollar-sign Expansion with an Expression

In this exercise we will use a copy of the straight table from the previous exercise 1 and change the second and third expression to make the table more flexible since hardcoded years in the expression would require maintenance of the expressions. In this case, the user is forced to select one year. Selections in the month dimension will not affect the table.

Try to find out advantages and disadvantages with using the function **Only**() compared to using a Modifier with an explicit field value.

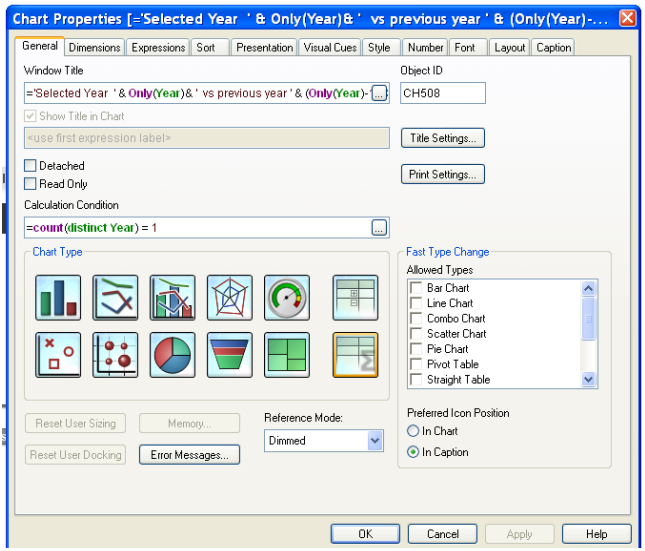
```
Expression =  Sum({$<Year= {$(=Only(Year))}, Month =  >} LineSalesAmount)
```

Selected Year 2007 vs previous year 2006, ex 2

DivisionName	sum (LineSalesAmount)	2007	2006	2007 vs 2006	2007 vs 2006 %
	3,058,936	3,058,936	1,153,955	**1,904,981**	**62.28%**
Europe	537,014	537,014	188,722	**348,292**	**64.86%**
North America	689,399	689,399	270,638	**418,761**	**60.74%**
Scandinavia	883,073	883,073	340,390	**542,683**	**61.45%**
South America	252,313	252,313	97,873	**154,440**	**61.21%**
X	697,137	697,137	256,331	**440,806**	**63.23%**

Do:

1 Make a **copy** of the straight table from the previous exercise 1.
2 On the **General tab** change the window **Title** to
='Selected Year ' & Only(Year)& ' vs previous year ' & (Only(Year)-1) & ', ex 2'

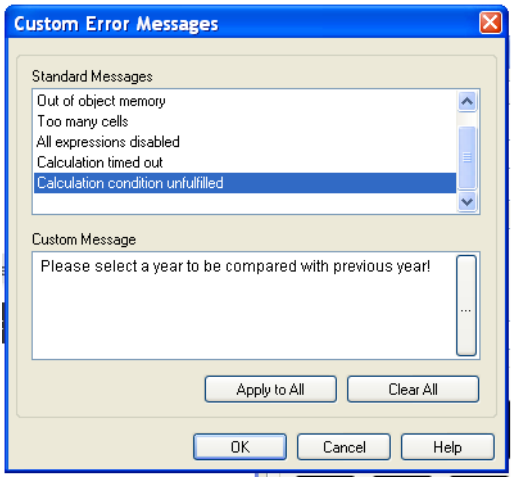


3 On the **General tab**, add a **Calculation Condition** to ensure that the user selects a Year to begin the comparison by entering the following into the **Calculation Condition box**

Count(distinct Year)=1

4 Click on the **Error Messages button** on the **General tab** and then on **Calculation Condition Unfulfilled** in the **Standard Messages** list.

5 **Type**: "Select a Year to compare with a previous year" in the Custom Message box and click OK.



6 On the expression tab change the second and third expression to:
Label 2-
=Only(Year)
Expression 2-
= sum(*{$<Year= {$(=Only(Year))}, Month = >}* **LineSalesAmount)**
Label 3-
=(Only(Year)-1)
Expression 3-
sum(*{$<Year = {$(=Only(Year)-1)}, Month= >}* **LineSalesAmount)**

7 Change the labels for expression four and five to make them show the selection of year.
Label 4-
=Only(Year) & ' vs ' & (Only(Year)-1)
Label 5-
=Only(Year) & ' vs ' & (Only(Year)-1)& ' %'

8 Click **Finish**

9 Click **OK** to save the table

10 **Save** the application

11 Leave the application open for the next exercise

Advanced Set Analysis Exercise 3

Dollar-sign Expansion with Expressions

In this exercise we will use a copy of the straight table from the previous exercise 2 and change the second and third expression to make the table always compare the latest year in selection to the previous year. In this exercise you will add two new columns to compare a month one year with the same month from the previous year. The month will be selected by using the **Max**() function.

Dollar-sign Expansion with an Expression, ex 3

DivisionName	sum (LineSalesAmount)	2010	2009	2010 vs 2009	2010 vs 2009 %	2010 Mar	2009 Mar
	13,321,238	535,098	3,562,331	-3,027,233	-565.73%	182,665	320,992
Europe	2,803,648	159,836	860,318	-700,481	-438.25%	48,897	78,027
North America	2,887,815	100,763	726,547	-625,785	-621.05%	29,959	49,849
Scandinavia	3,281,675	116,145	734,031	-617,885	-531.99%	60,238	77,029
South America	982,443	26,427	244,321	-217,894	-824.53%	8,698	20,479
X	3,365,657	131,927	997,114	-865,187	-655.81%	34,874	95,607

Try to find out advantages and disadvantages with using the function **Max**() compared to the function **Only**().

```
Expression =  sum({$<Year = {$(=Max(Year))}, Month = {$(=Month(Max(OrderDate)))} >} LineSalesAmount)
```

Do:

1 Make a **copy** of the straight table from the previous exercise 2.
2 On the **General tab** change the window Title to
=Dollar-sign Expansions with Expressions, exercise 3
3 On the **General tab** delete the **Calculation Condition**
4 On the expression tab change the second and third **labels** and **expression** to:
Label 2-
=Max(Year)
Expression 2-
= sum(*{$<Year= {$(=Max(Year))}, Month = >}* **LineSalesAmount**)
Label 3-
=(Max(Year)-1)
Expression 3-
sum(*{$<Year = {$(=Max(Year)-1)}, Month= >}* **LineSalesAmount**)

5 Change the **labels** for **expression** four and five to make them show the selection of year.

Label 4-

=Max(Year) & ' vs ' & (Max(Year)-1)

Label 5-

=Max(Year) & ' vs ' & (Max(Year)-1)& ' %'

6 Click **Finish**

7 Click **OK** to save the table

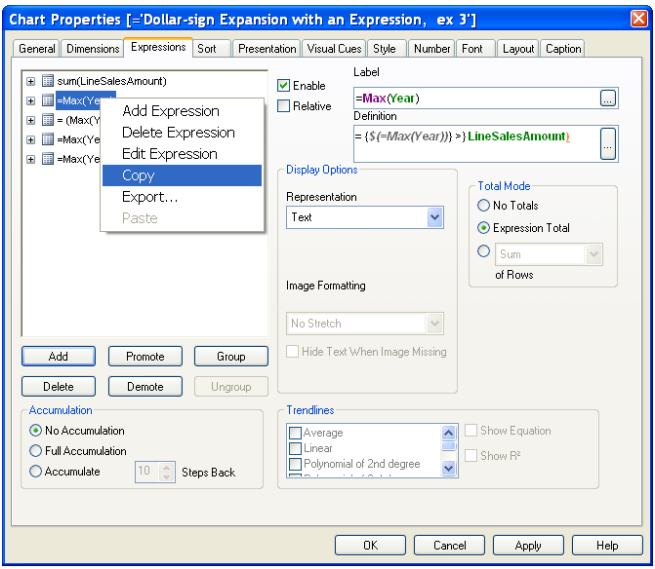
8 **Save** the application

9 Leave the application open for the next exercise

10 **Return** to the properties for the straight table, exercise 3

11 Time to add two columns to the table in exercise 3. Return to the Expression tab.

12 Right click on the second expression and **Copy** the expression.



13 Right click again and **Paste** the copied expression.

14 Change the label and the expression to:

Label 6

=Max(Year) & ' ' & Month(Max(OrderDate))

Expression 6:

sum(*{$<Year = {$(=Max(Year))}, Month = {$(=Month(Max(Order-Date)))} >}* **LineSalesAmount)**

15 Do the same copy and paste with the third expression and change the label and expression to:

Label 7

=(Max(Year)-1) & ' ' & Month(Max(OrderDate))

Expression 7:

Sum(*{$<Year = {$(=(Max(Year)-1))}, Month = {$(=Month(Max(OrderDate)))} >}* **LineSalesAmount)**

16 Click **OK** to save the table

17 **Save** the application

18 Leave the application open for the next exercise

Advanced Set Analysis Exercise 4

Dollar-sign Expansion using Variables

In this exercise we will use a copy the straight table from the previous exercise 3 and change the second and third expression to use a dynamic variable instead. The variable will be set to the current year and will be created by a date function.

It's important to clearly explain (for users) what the expression shows when you use Set Analysis, because Set Analysis overrides the normal QlikView association.

Try to find out advantages and disadvantages with using a variable compared to write a function **Max**(), for example?.

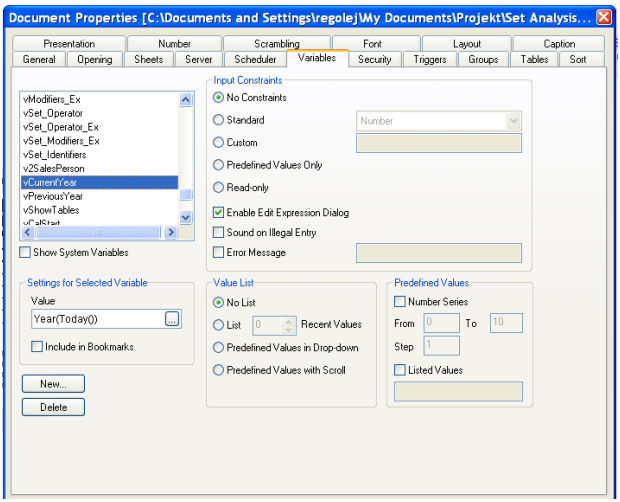
```
sum({$< Year = {$(=$(vCurrentYear))}> } LineSalesAmount)
```

Dollar-sign Expansion with a Variable, ex 4

DivisionName	sum (LineSalesAmount)	Current year 2010	Previous year 2009	Dev	Dev %		Current year 2010 Mar	Previous year 2009 Mar
	13,321,238	535,098	3,562,331	**3,027,233**	**84.98%**		182,665	320,992
Europe	2,803,648	159,836	860,318	**700,481**	**81.42%**		48,897	78,027
North America	2,887,815	100,763	726,547	**625,785**	**86.13%**		29,959	49,849
Scandinavia	3,281,675	116,145	734,031	**617,885**	**84.18%**		60,238	77,029
South America	982,443	26,427	244,321	**217,894**	**89.18%**		8,698	20,479
X	3,365,657	131,927	997,114	**865,187**	**86.77%**		34,874	95,607

Do:

1 Create a variable for the current year, previous year and one for the latest month in the selection.
2 From the menu **Settings** select **Document Properties**.
3 Click on the tab for **Variables**
4 Click on the button **New** to create a new variable.
5 Give the new variable the name **vCurrentYear** and click **OK**



6 In **Settings for selected Variable** =Year(Today())

7 Create the variable vPreviousYear and vCurrentMonth in the same way as vCurrentYear.

vPreviousYear = (Year(Today())-1)

vCurrentMonth = Month(Max(OrderDate))

8 Click **Ok**.

9 Make a **copy** of the straight table from the previous exercise 3.

10 On the **General tab** change the window Title to

=Dollar-sign Expansions with Variable, exercise 4

11 On the **expression** tab change the second and third **label** and **expression** to:

Label 2-

='Current year ' &$(vCurrentYear)

Expression 2-

Sum(*{$< Year = {$(=$(vCurrentYear))}> }* **LineSalesAmount)**

Label 3-

='Previous year ' & $(vPreviousYear)

Expression 3-

sum(*{$<Year = {$(=$(vPreviousYear))} >}* **LineSalesAmount)**

12 Change the labels for expression four to five to make them show the years from the variables.

Label 4-

='Current year ' &$(vCurrentYear) & ' vs ' & $(vPreviousYear)

Label 5-

='Current year ' &$(vCurrentYear) & ' vs ' & $(vPreviousYear) & ' %'

13 Change the labels for the expressions six and seven to make them show the years from the variables.

Label 6

='Current year ' &$(vCurrentYear) & ' ' & $(vCurrentMonth)

Expression 6:

sum(*{$<Year = {$(=$(vCurrentYear))}, Month = {$(=$(vCurrent-Month))} >}* **LineSalesAmount**)

Label 7

='Previos year ' & $(vPreviousYear) & ' ' & $(vCurrentMonth)

Expression 7:

sum(*{$<Year = {$(=$(vPreviousYear))}, Month = {$(=$(vCurrent-Month))} >}* **LineSalesAmount**)

14 Click **Finish**

15 **OK** to save the table

16 **Save** the application

Note: As an alternative to the process used in the exercise, the variables in this exercise could be created in the script as well.

Advanced Set Analysis Exercise 5

Search and Implicit Set Operator

Create a straight table showing sales per *Salesperson* for the current selection together with sales done by "Presley".

Make sure the Labels clearly explain what the column shows for the user, because the set analysis eliminate QlikView standard association.

```
Sum({$<SalespersonName += {"*Presley*"}>} LineSalesAmount)
```

Advanced Searcher and Implicit Set Operator, ex 5

SalesPerson	Sales for Current Selection	Sales for Current Selection in union with Presley
	8 026 207	8 535 663
Brolin, Helen	1 959 870	1 959 870
Callins, Joan	732 899	732 899
Carsson, Rob	2 625 611	2 625 611
Hendrix, Ingrid	637 644	637 644
Lindwall, Tom	2 070 182	2 070 182
Presley, Erik	0	509 456

Do:

1 Navigate to the **Advanced Searchers, Operators, P, E, Aggr** Sheet.
2 Right Click - **New Sheet Object | Chart | Straight Table**
3 On the **General Tab**, Name Window type: "**Searcher and Implicit Set Operator, ex 5**"
4 On the **Dimensions Tab**, add dimensions: **SalespersonName**
5 Change the dimension label to "**SalesPerson**".
6 On the **Expressions** Tab, add the expressions:
7 Create the following two **Expressions** using the **Labels** provided:

Label =Sales for Current Selection

Expression =Sum(LineSalesAmount)

Label =Sales for Current Selection in union with Presley

Expression = Sum(*{$<SalespersonName += {"*Presley*"}>}* **LineSalesAmount)**

Note: the portion of the expression, above, containing

{$<*SalespersonName* **= += {"*Presley*"}>}**

Sales for Current Selection in union with Presley by using the wildcard star (asterisk) *****

8 Set the Sort order to match the depiction, above, remembering that **SalespersonName** should be set to **Text**.
9 Continue to the **Number tab** and set all expressions to **Integer**.
10 Click **Finish**
11 **OK** to save the table
12 **Save** the application

After you finished this exercise try to achieve the same result in the first column with the expression **Sum**(*LineSalesAmont*) as the second column.

Check how different operators **+=**, **-=**, ***=**, **/=** in the second column expression change the result.

Advanced Set Analysis Exercise 6

Advanced Search using Set Modifiers with Element Functions P() and E()

Create a straight table showing (A) Sales for the current selection, but only those customers that ever have bought products that belongs to the *CategoryName* "Baby Clothes". Use element function **P**( ) which returns a list of possible customers; those that are implied by the selection "Baby Clothes" in the field CategoryName.

(B) Make another expression that shows the current selection, but only those customers that never have bought products from Supplier "SatSUMAs". Use element function(**E**) which returns a list of excluded Customers, those that are implied by the selection "SatSUMAs" in the field *SupplierName*.

(C) Add a third expression showing the current selection, but only those customer that ever have bought "Baby Clothes" and never have bought from "SatSUMAs", i.e. the intersection of expression (A) and (B).

A
```
Sum({$<CompanyName = P({1<CategoryName={'Baby Clothes'}>} CompanyName ) >}  LineSalesAmount)
```

B
```
Sum({$<CompanyName = E({1<SupplierName={'SatSUMAs'}>} CompanyName ) >}  LineSalesAmount)
```

C
```
Sum({$<CompanyName = P({1<CategoryName={'Baby Clothes'}>}  ) * E({1<SupplierName=
{'SatSUMAs'}>} CompanyName )>}  LineSalesAmount)
```

Advanced Searchers using Possible and Excluded, Ex 6

Customer	Sales for Current Selections	Customers buying Baby Clothes	Customers NOT buying from SatSUMAs	Customers buying Baby Clothes but NOT buying from SatSUMAs
	13 321 238	12 661 722	2 948 584	2 307 138
Alles Lusekofter	1 706	1 706	1 706	1 706
Art et Fashion	62 705	62 705	62 705	62 705
Aujourd'hui	13 125	13 125	13 125	13 125
Autokleider	28 817	0	28 817	0
Belgium Black Je...	73 392	0	73 392	0
Big Foot Shoes	77 938	77 938	77 938	77 938
Bobby Socks	13 895	13 895	0	0
Boleros	924 285	924 285	0	0

Do:

1 Navigate to the **Advanced Searchers, Operators, P, E, Aggr** Sheet.
2 Right Click - New Sheet Object | Chart | Straight Table
3 On the **General Tab, Name** Window type: "**Advanced Searcher using Possible and Exclusion**"
4 On the **Dimensions Tab**, add dimensions: **CompanyName**
5 Change the **dimension** label to "**Customer**".

6 On the **Expressions Tab**, create the following four **Expressions** using the **Labels** provided:

Label 1 =Sales for Current Selection

Expression 1 =sum(LineSalesAmount)

A Label 2 =Customer buying Baby Clothes

A Expression 2 =Sum(*{$<CompanyName = P({1<Category-RyName={'Baby Clothes'}>} CompanyName) >}* **LineSalesAmount)**

B Label 3 = Customers NOT buying from SatSUMAs

B Expression 3 = Sum(*{$<CompanyName = E({1<Supplier-Name={'SatSUMAs'}>} CompanyName) >}* **LineSalesAmount)**

C Label 4 = Customers buying Baby Clothes but NOT buying from SatSUMAs

C Expression 4 = Sum(*{$<CompanyName = P({1<Category-RyName={'Baby Clothes'}>}) * E({1<SupplierName={'SatSUMAs'}>}* **CompanyName)>} LineSalesAmount)**

7 Set the Sort order to match the depiction, above, remembering that **Customer** should be set to **Text**.

8 Continue to the **Number tab** and set all expressions to **Integer.**

9 Click **Finish**

10 Click **OK** to save the table

11 **Save** the application

Advanced Set Analysis Exercise 7

Set Modifiers with Element Functions P() and an Aggregated Function

In this exercise we will add an expression to the table we created in exercise 5.

This new column will show sales for per *SalesPerson*, but only the sales for those customers that ever have bought from salespeople that have a *SalesTitle* "Sales Representative". Use element function **P**( ) which returns a list of possible customers; those that are implied by the selection "Sales Manager" or "President" in the field *SalesTitle*.

```
Sum({$<CompanyName = P({1<SalesTitle={'Sales Manager', 'President'}>} CompanyName ) >}
LineSalesAmount)
```

Advanced Searcher and Implicit Set Operator, P, Aggr function, ex 5, 7

SalesPerson	Sales for Current Selection	Sales for Current Selection in union with Presley	Customers buying from Sales Manager and President
	13 321 238	13 321 238	5,246,857
Brolin, Helen	1 959 870	1 959 870	1,388,676
Callins, Joan	732 899	732 899	732,899
Carsson, Rob	2 625 611	2 625 611	1,984,220
Hendrix, Ingrid	637 644	637 644	631,606
Lindwall, Tom	2 070 182	2 070 182	0
Presley, Erik	509 456	509 456	509,456
Roll, Frank	2 043 316	2 043 316	0
Shine, Leif	1 902 031	1 902 031	0
Skoglund, Lennart	840 228	840 228	0

Do:

1 Navigate to the **Advanced Searchers, Operators, P, E, Aggr** Sheet.
2 Open the table from exercise 5, **Advanced Searcher and Implicit Set Operator**
3 On the **General Tab**, Name Window type: "**Advanced Searcher and Implicit Set Operator, P, Aggr function, ex 5 and 7**"
4 On the **Expressions Tab**, add the following third **Expressions** using the **Labels** provided:

Label 3 = Customers buying from Sales Manager and President

Expression 3 = Sum({$<*CompanyName = P({1<SalesTitle={'Sales Manager', 'President'}>} CompanyName) >}* **LineSalesAmount**)

5 Continue to the **Number tab** and set all expressions to **Integer.**
6 Click **Finish**
7 Click **OK** to save the table
8 **Save** the application

Advanced Set Analysis Exercise 8

Set Analysis using CONCAT function

Instead of writing a long list of field values in the set modifier you can use the **concat** () function to make the list of field values from a field, a function or a variable.

In our first exercise using **concat**() we will use month as a dimension and a date function together with **concat**() to build up a list of the last day of a month. We apply "distinct" to return a single value from every month, namely the last day.

Look at the first expression using the **concat**() function to only show the last day of the month by using the **monthend**() function.

```
sum({$<Emp_Date={"$(=concat(distinct monthend(Emp_Date),'','))"}>} Employees)
```

The expression below is a result of the expression above. You find all field values listed in the modifier, by the expression.

```
sum({$<Emp_Date={"10/31/2008","11/30/2008","12/31/2008","7/31/2008","8/31/2008","9/30/2008"}>} Employees)
```

In the table box to the left below, you can see the number of employees for each day and month. In the table to right you will see the number of employees only for the last day of each month.

Emp_Month	Emp_Day	No of Employees
Jan	1	78
Jan	2	80
Jan	3	81
Jan	4	84
Jan	5	82

Set analysis using Concat function, ex 8

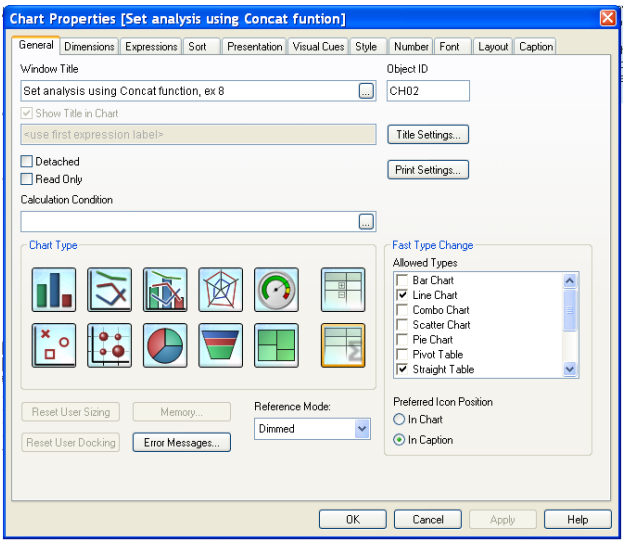
Emp_Month	No of Employees last day in the month
	391
Jan	109
Feb	138
Oct	20
Nov	47
Dec	77

Do:

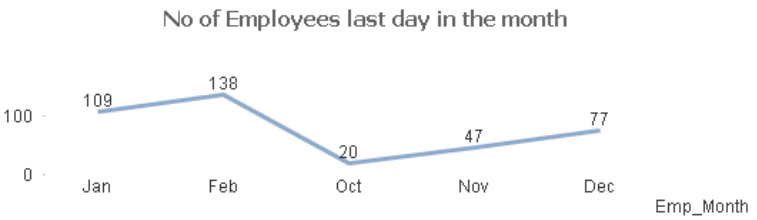
1 Navigate to the **Concat** Sheet
2 Right Click - **New Sheet Object | Chart | Straight Table**
3 On the **General Tab**, Name Window type: "**Set analysis using Concat function, ex 8**"
4 On the **Dimensions Tab**, add dimensions: **Emp_Month**
5 On **the Expressions Tab**, add the expressions:

6 Create the following **Expressions** using the **Labels** provided:
Label =No of Employees last day in the month
Expression =sum(*{$<Emp_Date={"$(=concat(distinct monthend(Emp_Date),'',''))"}>}* **Employees)**

7 Return to the **General Tab**, **Fast Type Change**, check mark **Line Chart** and **Straight Table**



8 Click **OK** to save the table
9 **Save** the application



Advanced Set Analysis Exercise 9,

Set Analysis using Concat() and Data Island

From the selection in Data Island field we build up a list by using the **Concat**() function.

```
Sum( {<SalespersonName={"$(=concat(distinct DataIsland_1.SalespersonName,"',"'))"}>} LineSalesAmount)
```

In the expression below, you find all field values selected from the data island listed in the modifier.

```
Sum( {<SalespersonName={"Callins, Joan","Carsson, Rob","Roll, Frank"}>} LineSalesAmount)
```

In this exercise you will create a table showing Sales per salesperson per Year. The first column will show Sales by the current selection and the second column shows sales for these salespeople selected in the data island *DataIsland_1.SalespersonName* and current selection, disregarding the selection in the ordinary *Salesperson* field.

DataIsland_1.SalespersonName
Brolin, Helen
Callins, Joan
Carsson, Rob
Hendrix, Ingrid
Lindwall, Tom
Presley, Erik
Roll, Frank
Shine, Leif
Skoglund, Lennart

Sales per Salesperson using Concat and data island

Year	Sales for Current Selections	Data for : 'Callins, Joan','Carsson, Rob','Ro...
	13 321 238	5 401 826
2005	119 793	47 619
2006	1 153 955	552 215
2007	3 058 936	1 382 050
2008	4 891 125	1 990 890
2009	3 562 331	1 274 716
2010	535 098	154 335

Do:

1 Navigate to the **Concat() and Data Island** Sheet
2 Open **Properties** for the existing table Sales per Year.
3 On the **General Tab**, Name Window type: "**Sales per Salesperson using Concat and a Data Island, ex 9**"

4 On the **Expressions Tab**, add the following **Expressions** using the **Labels** provided:

Label = ='Data for : '& concat(distinct DataIsland_1.Salesperson-Name,'",","')

Expression = Sum({<*SalespersonName={"$(=concat(distinct DataIsland_1.SalespersonName,'",","))"}*>} **LineSalesAmount**)

5 Continue to the **Number tab** and set all expressions to **Integer.**

6 Click **OK** to save the table

7 **Save** the application

Advanced Set Analysis Exercise 10,

Set Analysis using concat() and Data Island in a variable

Instead of writing (A) the **concat**() function in the expression it is possible to create (B) a variable for the **concat**() function and then use the variable in (C) the expression.

A

```
Sum( {<SalespersonName={"$(=concat(distinct DataIsland_1.SalespersonName,"","))"}>} LineSalesAmount)
```

B

```
vSalesPerson = chr(39)&concat(distinct DataIsland_1.SalespersonName,chr(39)&chr(44)&chr(39))&chr(39)
```

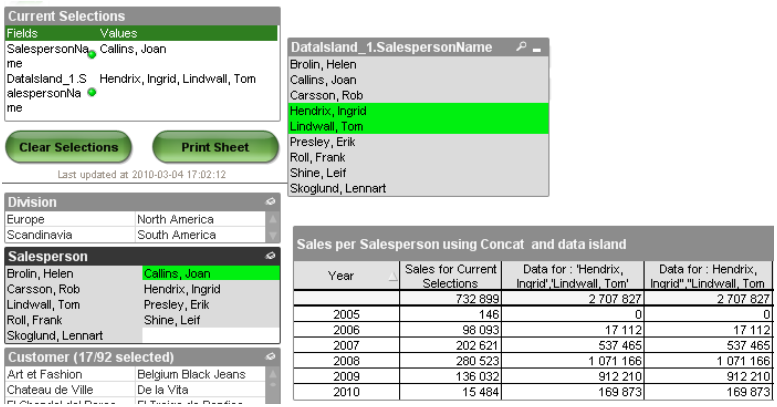
chr(39) = ' and chr(44) = , (comma)

C.

```
Sum( {<SalespersonName={$(vSalesPerson)}>} LineSalesAmount)
```

In Column 3 in the table below, the **concat** function (A) is written within the expression. Column 4 shows the same result but uses the variable (B) in the expression (C).

Question: When or why is a variable preferable?



Add the third column to the existing table using a variable in the expression. Start by creating the variable.

Do:

1 Navigate to the **Concat() and Data Island** Sheet
2 Create a variable **vSalesperson** using the **concat()** function to build up a list from the data island field **DataIsland_1.SalespersonName**.
3 From the menu **Settings** select **Document Properties**.
4 Click on the button **New** to create a new variable.
5 Give the new variable the name **vSalesPerson** and click **OK**
6 In **Settings for vSalesperson**

= vSalesPerson = chr(39)&concat(distinct DataIsland_1.Salesperson-Name,chr(39)&chr(44)&chr(39))&chr(39)

Note: Using the **chr** function in conjunction with the ASCII number for a character in the **vSalesPerson** variable insures that the correct character will be returned, in this case the apostrophe (') and the comma (,) regardless of the cmoputing environment.

7 Open **Properties** for the existing table **Sales per Year.**
8 On the **General Tab**, Name Window type: "**Sales per Salesperson using Concat and a data island, ex 9, 10**"
9 On the **Expressions** Tab, add the following **Expressions** using the **Labels** provided:

Label = ='Data for : '&(vSalesPerson)

Expression = Sum(*{<SalespersonName={$(vSalesPerson)}>}* **LineSalesAmount)**

10 Continue to the **Number tab** and set all expressions to **Integer.**
11 Click **OK** to save the table
12 **Save** the application

Advanced Set Analysis Exercise 11

Set Analysis using concat() and Data Island in a variable

Add two columns to the table from the previous exercise 10, showing (D) Sales for the current selection and sales of *CategoryNames* equal to the selection in *DataIsland_2.CategoryName*.

(E) Make another expression that shows the current selection and sales from **SupplierName** equal to the selection in *DataIsland_3.SupplierName*.

D `Sum( {<CategoryName = {$(vCategoryName)}>} LineSalesAmount)`

E `Sum( {<SupplierName = {$(vSupplierName)}>} LineSalesAmount)`

DataIsland_1.SalespersonName
Brolin, Helen
Callins, Joan
Carsson, Rob
Hendrix, Ingrid
Lindwall, Tom
Presley, Erik
Roll, Frank
Shine, Leif
Skoglund, Lennart

DataIsland_2....
Baby Clothes
Childrens Clothes
Mens Clothes
Mens Footwear
Sportswear
Swimwear
Womens Clothes
Womens Footwear

DataIsland_3....
Big L
Cangaroo Shoes
ABC
Asin Fashion Ltd Co
Austerlich
Bar Åkeri
Der Bahnhof
Dressed for Succes
Executive Clothing GMBH

Sales per Salesperson using Concat and data island

Year	Sales for Current Selections	Data for : 'Brolin, Helen','Callins,...	Data for : Brolin, Helen","Callins,...	Data for : 'Childrens Clothes','Mens...	Data for : 'Big L','Cangaroo Shoes'
	13 321 238	13 321 238	13 321 238	3 612 712	1 574 712
2005	119 793	119 793	119 793	31 445	7 650
2006	1 153 955	1 153 955	1 153 955	287 571	160 736
2007	3 058 936	3 058 936	3 058 936	811 738	366 197
2008	4 891 125	4 891 125	4 891 125	1 324 833	592 951
2009	3 562 331	3 562 331	3 562 331	1 016 517	393 948
2010	535 098	535 098	535 098	140 609	53 230

Do:

1 Navigate to the **Concat() and Data Island** Sheet
2 Create a variable **vCategoryName** using the concat() function to build up a list from the data island field **DataIsland_2.CategoryName**.
3 From the menu **Settings select Document Properties**.
4 Click on the button **New** to create a new variable.
5 Give the new variable the name **vCategoryName** and click **OK**
6 In Settings for vCategoryName = vCategoryName = chr(39)&concat(distinct DataIsland_2.CategoryName,chr(39)&chr(44)&chr(39))&chr(39)

7 Open **Properties** for the existing table **Sales per Year.**
8 On the **General Tab**, Name Window type: "**Sales per Salesperson using Concat and a data island, ex 9, 10**"
9 On the **Expressions Tab**, add the following **Expressions** using the **Labels** provided:

Label = ='Data for : '&(vCategoryName)

Expression = Sum(*{<CategoryName={$(vCategoryName)}>}* **LineSalesAmount)**

Label = ='Data for : '&(vSupplierName)

Expression = Sum(*{< SupplierName ={$(vSupplierName)}>}* **LineSalesAmount)**

10 Continue to the **Number tab** and set all expressions to **Integer.**
11 Click **OK** to save the table
12 **Save** the application

Advanced Set Analysis Exercise 12

Advanced Searches using Set Modifiers with Element Functions P() and E() and Data Islands

In this exercise we will use the straight table from exercise together with the variables and Data Islands we already created and use the combination to make the new table dynamic.

The new table will show (A) Sales for the current selection, but only for those customers that have ever purchased products from the *CategoryName* selected from the *DataIsland_2.CategoryName*. Use element function **P**( ) which returns a list of possible customers; those that are implied by the selection in the field *DataIsland_2.CategoryName*.

(B) The next expression shows the current selection, but only for those customers that have never purchased products from the *Supplier* selected from the *DataIsland_3.SupplierName*. Use element function(**E**) which returns a list of excluded Customers, those that are implied by the selection in the field *DataIsland_3.SupplierName*.

(C) The third expression shows the intersection of expression (A) and (B).

A
```
Sum({$<CompanyName = P({1<CategoryName = {$(vCategoryName)}>} CompanyName ) >} LineSalesAmount)
```

B
```
Sum({$<CompanyName = E({1<SupplierName={$(vSupplierName)}>} CompanyName ) >} LineSalesAmount)
```

C
```
Sum({$<CompanyName = P({1<CategoryName={$(vCategoryName)}>} ) * E({1<SupplierName={$(vSupplierName)}>} CompanyName )>} LineSalesAmount)
```

Possible and Excluded selected from Data Island, ex 12

Customer	Sales for Current Selections	Customers buying 'Mens Clothes'	Customers NOT buying from 'Der Bahnhof'	Customers buying 'Mens Clothes' but NOT buying...
	13 321 238	12 844 133	3 669 331	3 265 279
Art et Fashion	62 705	62 705	0	0
Aujourd'hui	13 125	13 125	13 125	13 125
Autokleider	28 817	28 817	28 817	28 817
Belgium Black Jeans	73 392	73 392	73 392	73 392
Big Foot Shoes	77 938	77 938	77 938	77 938
Bobby Socks	13 895	13 895	13 895	13 895
Boleros	924 285	924 285	0	0
Bond Ltd	306 510	306 510	0	0

Do:

1 Navigate to the **Advanced Searchers, Operators, P, E,** Sheet.
2 **Copy** the table **Advanced Searchers** using **Possible and Excluded, Ex 6**

3 Navigate to the **Concat()** and **Data Island Sheet**.
4 **Paste** the copied table and open the properties for the table
5 On the **General Tab**, Name Window type: **"Advanced searcher using data island and variable, ex 12"**
6 Create the following three **Expressions** using the **Labels** provided:

A Label 2 = ='Customers buying ' & vCategoryName

A Expression 2 = Sum(*{$<CompanyName = P({1<CategoryName = {$(vCategoryName)}>} CompanyName) >}* **LineSalesAmount**)

B Label 3 = ='Customers NOT buying from ' & vSupplierName

B Expression 3 = Sum(*{$<CompanyName = E({1<Supplier-Name={$(vSupplierName)}>} CompanyName) >}* **LineSalesAmount**)

C Label 4 = ='Customers buying ' & vCategoryName & ' but NOT buying from ' & vSupplierName

C Expression 4 = Sum(*{$<CompanyName = P({1<CategoryName={$(vCategoryName)}>}) * E({1<SupplierName={$(vSupplierName)}>} CompanyName)>}* **LineSalesAmount**)

7 Continue to the **Number tab** and set all expressions to **Integer.**
8 Click **Finish**
9 Click **OK** to save the table
10 **Save** the application

Advanced Set Analysis Exercise 13

Aggregated functions

In this exercise we will add two expressions to the straight table from exercise 5 and 7. The first expression is with a hardcoded modifier. With the second expression, we will make it more dynamic when we use variables.

The added forth column in this table will show (**A**) Sales for the current selection and the salespersons who sold for more than $100,000 during year 2007.

(**B**) The fifth expression includes the **max**(Year) function and a variable for the *SalesAmont*. Notice, too, that the aggregation field is changed from *SalespersonName* to *CompanyName* (Customer), in order to show the current selection and Customers who bought for more than the value of the variable *SalesAmount* during the latest year in the selection.

A `sum({$< SalespersonName = {"=Sum({1<Year = {2007}>} LineSalesAmount)> 100000"} >} LineSalesAmount)`

B `sum({$< CompanyName = {"=Sum({1<Year= {$(=Max(Year))}>} LineSalesAmount)> $(vSalesAmount)"} >} LineSalesAmount)`

Advanced Searcher and Implicit Set Operator, P, Aggr function, ex 5, 7, 13

SalesPerson	Sales for Current Selection	Sales for Current Selection in union with Presley	Customers buying from Sales Manager and President	Salesperson who sold for more than than 100,000 during 2007	Customer who bought more than 100000 during 2010
	13 321 238	13 321 238	5,246,857	12,683,594	1,431,752
Brolin, Helen	1 959 870	1 959 870	1,388,676	1,959,870	0
Callins, Joan	732 899	732 899	732,899	732,899	0
Carsson, Rob	2 625 611	2 625 611	1,984,220	2,625,611	0
Hendrix, Ingrid	637 644	637 644	631,606	0	0
Lindwall, Tom	2 070 182	2 070 182	0	2,070,182	720,726
Presley, Erik	509 456	509 456	509,456	509,456	0
Roll, Frank	2 043 316	2 043 316	0	2,043,316	0
Shine, Leif	1 902 031	1 902 031	0	1,902,031	711,026
Skoglund, Lennart	840 228	840 228	0	840,228	0

Do:

1 Navigate to the **Advanced Searchers, P, E, Concat, Aggr** Sheet.
2 Open the **properties** for the table from exercise 5 and 7, **Advanced Searcher and Implicit Set Operator,**
3 On the **General Tab**, Name Window type: "**Advanced Searcher and Implicit Set Operator, P, Aggr function**, ex 5, 7, 13"

4 Create the following two **Expressions** using the **Labels** provided:

Label 4 = Salesperson who sold for more than 100,000 during 2007

Expression 4 = sum(*{$< SalespersonName = {"=Sum({1<Year = {2007}>}*

LineSalesAmount)> 100000"} >} **LineSalesAmount)**

Label 5 = ='Customer who bought more than ' & vSalesAmount & ' during ' & Max(Year)

Expression 5 = =sum(*{$< CompanyName = {"=Sum({1<Year= {$(=Max(Year))}>} LineSalesAmount)> $(vSalesAmount)"} >}* **LineSalesAmount)**

5 Continue to the **Number** tab and set all expressions to **Integer.**

6 Click **Finish**

7 **OK** to save the table

8 **Save** the application

Extending Set Analysis: Exercise 14

Do:

1 Check SA Time Report 1 Sheet in Solution application for Set Analysis Course.qvw.
2 This sheet contains two reports with set expressions to compare different time periods. This is the type of example that could form the basis of future applications to solve other business problems, as needed, or with small modifications.

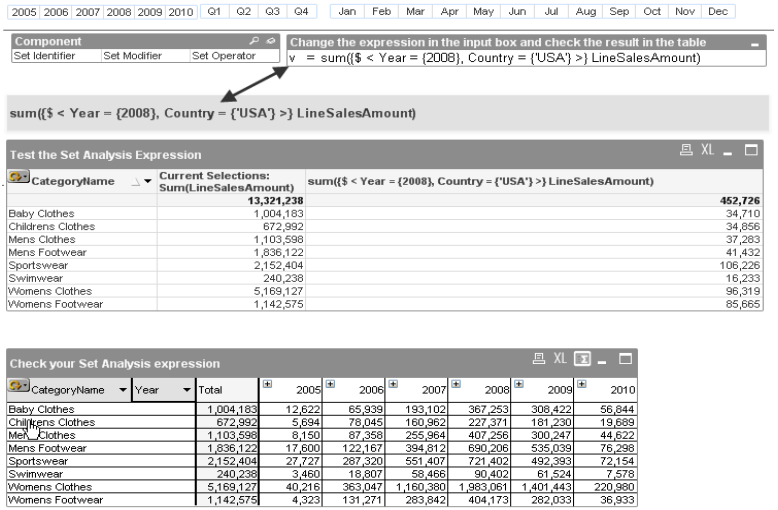
Extending Set Analysis: Exercise 15

Do:

1 Check SA Time Report 2 Sheet in Solution application for Set Analysis Course.qvw.
2 This sheet shows two different ways of how to use Dynamic Set Analysis for comparing different periods in reports.
3 The first is by using variables for time functions and the second is by creating variables for a complete set expression.

Debugging Set Analysis: Exercise 16

Evaluate the Set Expression in the Basic Component Sheet



Do:

1 Navigate to the **Basic Component** Sheet.
2 Clear the list box **Component**
3 Type or paste the **Set expression** you would like to explore i n the input box
4 Check the result in the second expression column in the **Straight table**
5 Compare the result with the first expression column
6 Change the **dimension** in the **cyclic group**, if necessary or depending on the set expression, to an appropriate **dimension**
7 Make selections in the list boxes to evaluate if the resolution works properly
8 Leave the application open for the next exercise

Error Searching Set Analysis: Exercise 17

Set Analysis Step-by-step

Create an advanced set analysis expression step by step.
If a Set analysis expression does not work the way you planned it, try to create an expression without the Set Analysis and create the result wanted by making the relevant selections in the layout. By doing so, you will get an outline for your set analysis and you will have a result to compare your Set Analysis to.

When building the Set Analysis, especially if it is a complex Set Analysis with several smaller parts put together, try to break it down into the smaller pieces and make sure that each piece works as intended before putting the entire Set Analysis together piece by piece.

Let's create a straight table showing the sales during the current year, disregarding any month selection, for all customers buying clothes, but only for those that do not belong to the Europe division.

Do:

1 Navigate to the **Basic Component Sheet**.
2 Clear the list box **Component**
3 Check what modifiers you will need to build the set expression.

a.) Current **Year**.

b.) Disregarding **Month** selections.

c.) All sales for **customers** who ever have bought from **Categories** including the text string ***Clothes***

d.) Exclude **customers** from the Europe **division**

4 Make the selections manually in the list boxes to evaluate if it's possible to do the selection.

Test the Set Analysis Expression

CompanyName	Current Selections: Sum(LineSalesAmount)	sum({1}LineSalesAmount)
	226,523	**13,321,238**
Alles Lusekofter	0	1,706
Art et Fashion	0	62,705
Aujourd'hui	0	13,125
Autokleider	0	28,817
Belgium Black Jeans	2,230	73,392
Big Foot Shoes	5,760	77,938
Bobby Socks	0	13,895
Boleros	23,383	924,285
Bond Ltd	7,789	306,510

5 **Save** the selection as a **bookmark**.
6 **Clear** the selection
7 Start with the basic expression without any set functions **Sum**(*LineSalesAmount*)
8 Add the basic set expression syntax Sum({$<>} LineSalesAmount)
9 Type the first part of the Set expression you would like to explore from in the input box. Start with Sum(*{$<Year = {2010} >}* LineSalesAmount)
10 Make the same selection in the list box and make sure that the result from the first and second column are the same
11 Change the specified year to $-sign variable or function and check that the results are the same. Do this by replacing **2010** in the previous expression with the $(=Year(Today())) or a variable for the current year *{$(=$(vCurrentYear))}*.

```
=Sum({$<Year = {$(=Year(Today()))} >} LineSalesAmount)
```

```
=Sum({$< Year = {$(=$(vCurrentYear))}> } LineSalesAmount)
```

12 Check the result of the set analysis expression in the chart to confirm the right result is achieved.
13 Add the next modifier, Month, which must be set to not be effected by any selection in the Field Month.
Sum*({$<Year = {$(= $(vCurrentYear))}, Month= >}* **LineSalesAmount**)
14 Add the an advanced searcher using wildcards to select all Categories including the text string
Clothes = **CategoryName = "*Clothes*"**

```
=Sum({$ < Year = {$(= $(vCurrentYear))}, Month=, CategoryName= {"*Clothes*"} > } LineSalesAmount)
```

So far, we made the set expression to select Sales for the current year disregarding selection in month and for categories containing the string *Clothes*. The last piece of the modifier expression will give us a set of

customers that have ever made a purchase, disregarding purchase from other Categories or which year they made the purchase. Since we want the total sales for those customers, we need to use the element function **P** to select those customers that have ever bought from Category *Clothes*.

15 Check the aggregated element function in a separate expression. There are different ways to keep your existing expression. For example Copy your expression and paste it, and edit the copied expression or copy the expression and paste it within the same expression, but comment one of the expressions and edit the other one.

```
=Sum({$ < CompanyName = {P({1<CategoryName= {"*Clothes*"} > })} > } LineSalesAmount)

//Sum({$ < Year = {$(= $(vCurrentYear))}, Month=, CategoryName= {"*Clothes*"}> } LineSalesAmount)
```

```
Sum({$ < CompanyName = {P({1<CategoryName= {"*Clothes*"} > })} > } LineSalesAmount)
```

Test the Set Analysis Expression

CompanyName	Current Selections: Sum(LineSalesAmount)	Sum({$ < CompanyName = {P({1<CategoryName= {"*Clothes*"} > })} > } LineSalesAmount)...
	13,321,238	13,314,661
Kohl Industries AG	2,517	0
Leningrad Cowboys Shop	4,061	0
Da Santho Cosmethia	565	565
Los Sombreros Gigantes	982	982
Alles Lusekofter	1,706	1,706
Fawtly Towers	2,158	2,158
For The Dark Night	2,274	2,274
SSS-Sport Shoes Store	4,594	4,594
Pulp Toxedos	4,683	4,683

16 Change the cyclic group, if necessary or depending on the set expression, to an appropriate dimension.

17 Check the result. By changing the sort order of the last column we can easily see the customers on the two first lines that didn't buy any clothes. When we manually select these two customers we will see that they never bought any clothes.

18 Before we apply the **P** Element function we check the **E** element function to exclude all customers that belongs to *DivisionName* = Europe. We make sure our set expression will fulfill this statement.

```
=Sum({$ < CompanyName = {E({1<DivisionName ={'Europe'}>})} > } LineSalesAmount)
```

19 When we check the set expression on the dimension *DivisionName* we can point out that there are no sales in Europe in the right column which represent the set expression.

```
Sum({$ < CompanyName = (E({1<DivisionName ={'Europe'}>}))> } LineSalesAmount)
```

Test the Set Analysis Expression

DivisionName	Current Selections: Sum(LineSalesAmount)	Sum({$ < CompanyName = (E({1<DivisionName ={'Europe'}>}))> } LineSalesAmount)...
	13,321,238	**11,238,827**
Europe	2,082,411	0
South America	1,397,155	1,397,155
North America	3,204,310	3,204,310
Scandinavia	3,280,563	3,280,563
X	3,356,799	3,356,799

20 In the next step you will apply the intersection of the two aggregated element functions into the previous expression. Replace

*CategoryName = (“*Clothes*”)*

with

*CompanyName = (P({1<CategoryName= {“*Clothes*”} > })) * (E({1<DivisionName ={'Europe'}>}))*

21 The final set expression will look as follows:

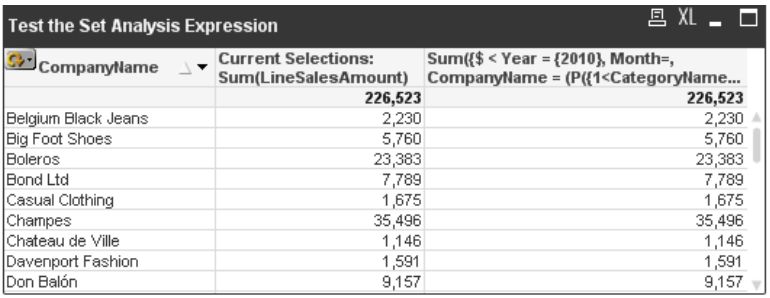
Sum(*{$ < Year = {$(= $(vCurrentYear))}, Month=, CompanyName = (P({1<CategoryName= {“*Clothes*”} >})) * (E({1<DivisionName ={'Europe'}>}))> } LineSalesAmount*)

```
Sum({$ < Year = {$(= $(vCurrentYear))}, Month=, CompanyName = (P
({1<CategoryName= {"*Clothes*"} >})) * (E({1<DivisionName ={"Europe"}>}))> }
LineSalesAmount)
```

Test the Set Analysis Expression

CompanyName	Current Selections: Sum(LineSalesAmount)	Sum({$ < Year = {2010}, Month=, CompanyName = (P({1<CategoryName...
	13,321,238	**374,684**
Alles Lusekofter	1,706	0
Art et Fashion	62,705	0
Aujourd´hui	13,125	311
Autokleider	28,817	0
Belgium Black Jeans	73,392	3,466
Big Foot Shoes	77,938	11,102
Bobby Socks	13,895	0
Boleros	924,285	57,937
Bond Ltd	306,510	13,326

22 Compare your set expression with the Bookmark you created in the beginning of the exercise.



Test the Set Analysis Expression

CompanyName	Current Selections: Sum(LineSalesAmount)	Sum({$ < Year = {2010}, Month=, CompanyName = (P({1<CategoryName...
	226,523	226,523
Belgium Black Jeans	2,230	2,230
Big Foot Shoes	5,760	5,760
Boleros	23,383	23,383
Bond Ltd	7,789	7,789
Casual Clothing	1,675	1,675
Champes	35,496	35,496
Chateau de Ville	1,146	1,146
Davenport Fashion	1,591	1,591
Don Balón	9,157	9,157