

NIL VOLENTIBUS ARDUUM

REPORT

BUILDER

AN

INTRODUCTION

Inhoud

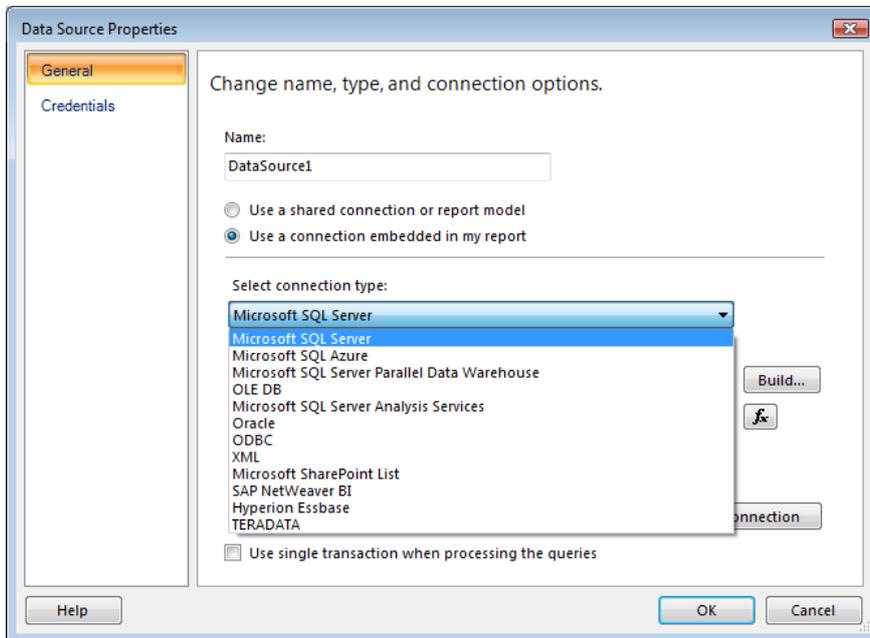
1	Getting the Data	1
1.1	Introduction	1
1.2	Report Builder Gui	1
1.3	Data Sources with the Wizard.....	2
1.4	Manual Data Set Construction.....	6
1.5	Data Construction Using The Query Designer	9
1.6	Data Regions.....	12
1.7	More Dataset Properties	13
1.7.1	Fields.....	14
1.7.2	Options	17
1.7.3	Filters	19
1.7.4	Complex formulas	19
1.7.5	Parameters	20
1.7.6	Expressions that refer to multivalue parameters	26
1.7.7	Parameter and Ranges	27
1.8	Sorting the data	28
1.8.1	Apply Sorting to the Column Headers.....	28
1.8.2	Dynamic Sorting Using Parameters	29
1.8.3	Dynamic Sorting through the Dataset.....	32
2	Designing the Report.....	35
2.1	Designing a Report Manually	35
2.2	Dragging fields directly to the report.....	36
2.3	Data Regions.....	37
2.3.1	Introduction	37
2.3.2	Table	37
2.3.2.1	Introduction.....	37
2.3.2.2	Adding a Table to Display Detail Data	38
2.3.2.3	Adding Totals for Detail Data	39
2.3.2.4	Adding Row Groups to a Table	40
2.3.2.5	Adding Totals to Row Groups.....	42
2.3.2.6	Removing or Hiding Detail Rows	42
2.3.2.7	Table properties	43
2.3.3	Matrix.....	46
2.3.3.1	Introduction.....	46
2.3.3.2	Adding a Matrix to Your Report.....	46
2.3.3.3	Adding a Parent Group or Child Group to a Matrix	48
2.3.3.4	Adding an Adjacent Group to a Matrix.....	49
2.3.4	List	50
2.3.4.1	Introduction.....	50
2.3.4.2	Adding a List to Your Report.....	50
2.3.4.3	Displaying Data in a Free-form Layout	51
2.3.4.4	Displaying Data with One Level of Grouping.....	52
2.4	Header & Footer.....	53
3	Formatting the Report.....	55
3.1	Cell formatting	55
3.1.1	To Allow a Text Box to Grow or Shrink	55
3.1.2	Underlining.....	55
3.1.3	Numeric/currency	56
3.1.4	Alternate row coloring.....	56
3.1.5	Displaying Checkboxes Instead of True/False	57
3.1.5.1	Introduction.....	57
3.1.5.2	Method 1: Images.....	57
3.1.5.3	Method 2: The Wingdings Font	59
3.1.6	The Indicator.....	60
3.2	Page formatting	64
3.2.1	Margins	64
3.2.2	Page orientation	66
3.2.3	Page numbering	66
3.2.4	Built-in fields.....	69
3.3	Add a Boolean Parameter for Conditional Visibility	71
4	Adding Groups.....	72
4.1	What Makes a Group?.....	72
4.2	When Do We Create Groups?.....	72
4.3	How Can We Modify a Group?.....	72

4.4	How are Groups Organized?.....	72
4.5	What Types of Groups are Available per Data Region?	73
4.6	Groups in a Tablix Data Region: Details, Row, and Column Groups	73
4.7	Understanding Group Membership for Tablix Cells	74
4.8	Grouping options	74
5	Calculating Totals.....	81
5.1	To group data in a report.....	81
5.2	To add totals to a report	82
5.3	To add a daily total to a report	82
5.4	To add a grand total to a report	83
6	Calculating Running Totals.....	84
6.1	Introduction	84
6.2	Resetting the RunningValue per Group	85
6.3	RunningValue in a condition.....	85
6.4	Conditional Running Total	86
6.5	Example of conditional, parameterized Running Total	86
7	Adding Graphs	88
7.1	Introduction	88
7.2	Designing a Chart	88
7.3	Similarities to a Matrix	90
7.4	Adding Data to the Chart	91
7.5	Category and Series Groups in a Chart.....	91
7.6	Dataset Considerations Before Creating a Chart	92
7.7	Best Practices When Displaying Data in a Chart	92
7.8	Aggregating Values from a Data Field on the Chart.....	92
8	Exporting the Data.....	94
9	Complex Exercise	95
9.1	Information analysis	95
9.1.1	The information we want	95
9.1.2	What it should look like	96
9.2	Creating the report in Report Builder.....	96
9.2.1	The structure and the numbers	96
9.2.2	Formatting.....	102
9.2.3	Graph.....	105
9.3	Linking scheme	107

1 Getting the Data

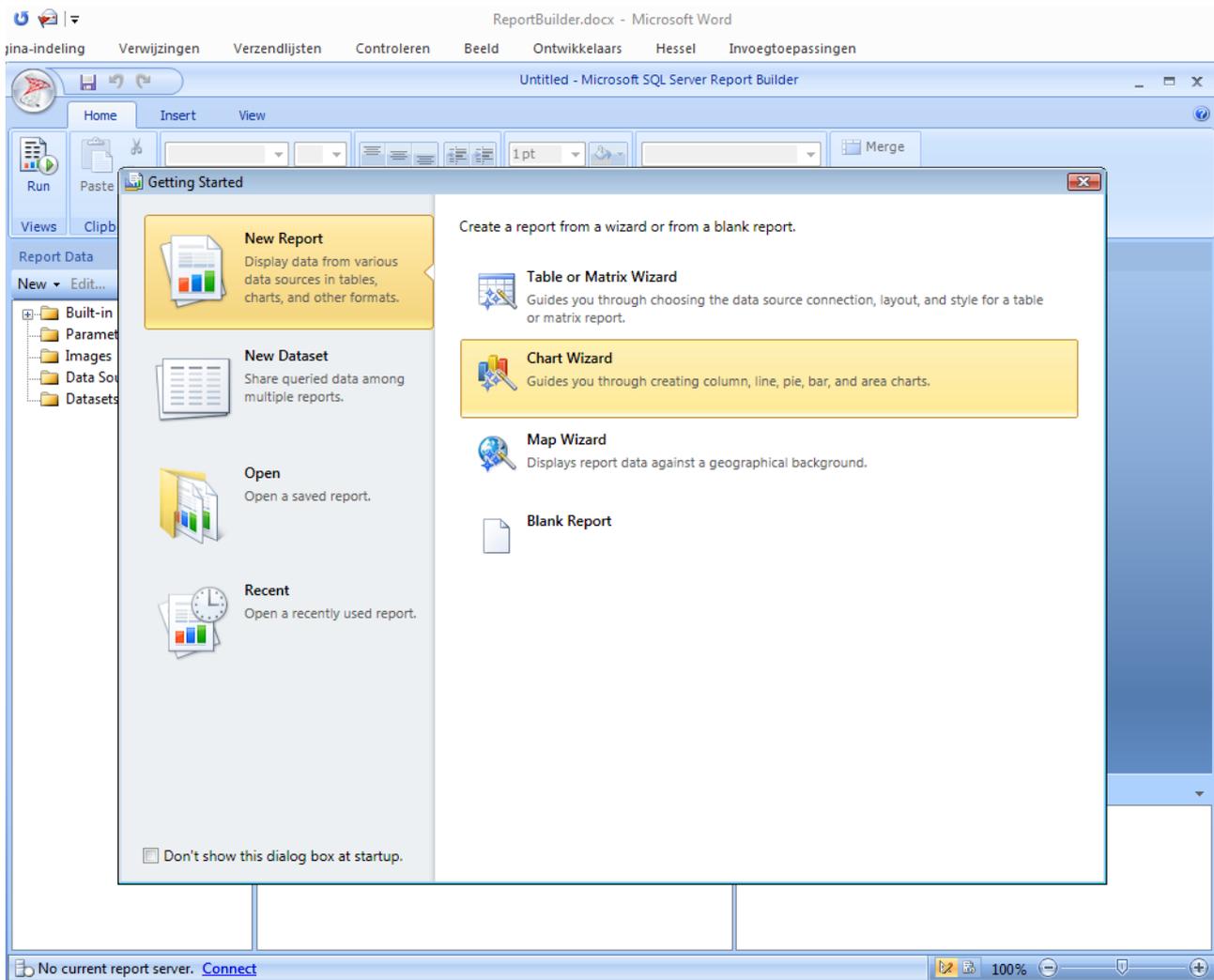
1.1 Introduction

With Report Builder you can connect to all types of databases. In this course material we will concentrate on SQL Server.



1.2 Report Builder Gui

- Start the Report Builder Gui.



Each report requires a data source. Data sources can either be shared or embedded.

- Shared sources are stored on the Report Server and can be used in any given reports given appropriate permissions.
- Embedded data sources are local to the specific report. The connection information is contained only in that report and cannot be used for other reports. If an embedded data source is required in another report, the connection must be specified anew there as well.

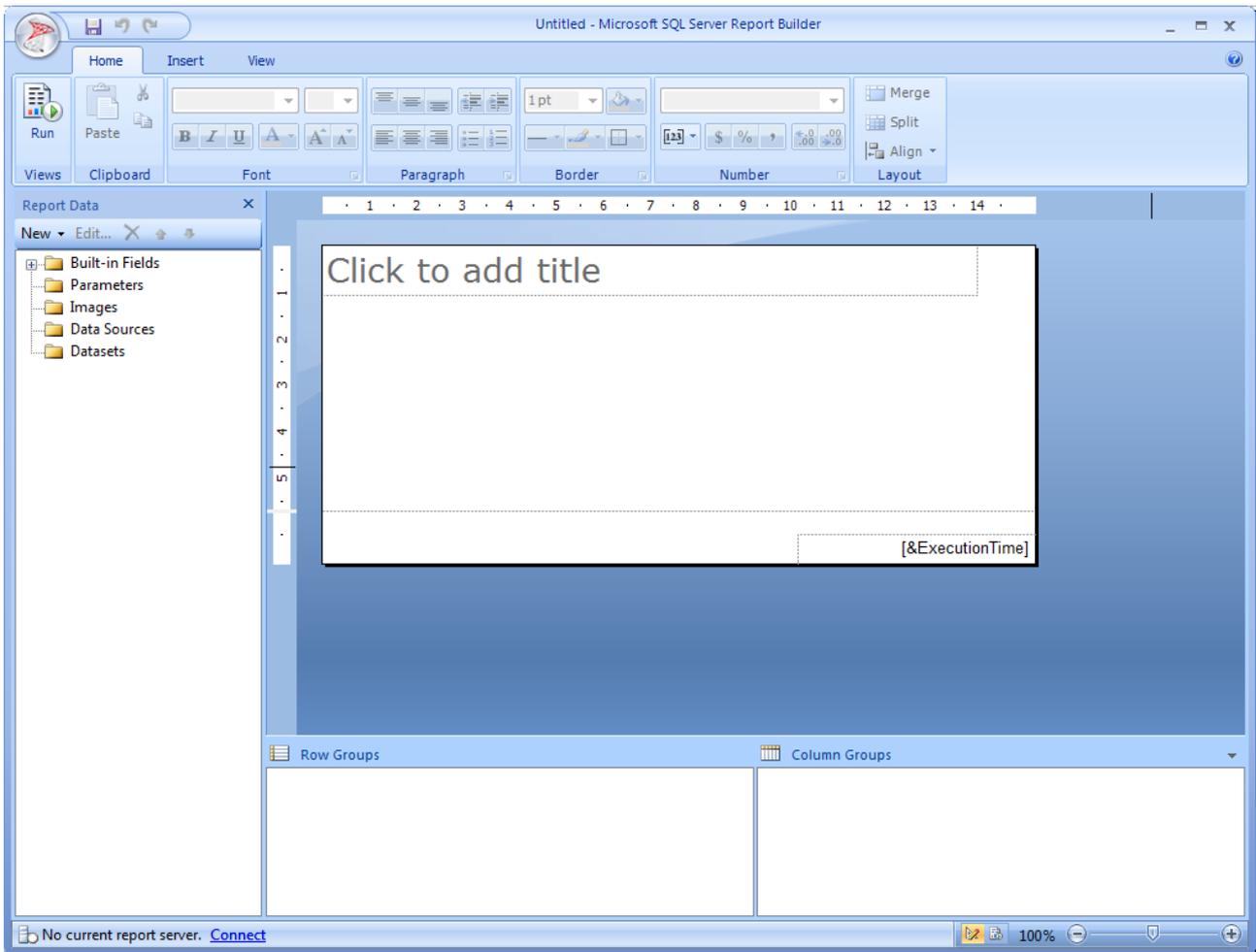
If a shared data source is not available, you have three options:

- Use an embedded data source.
- Go to the Report Manager and create a new shared Data Source.
- Ask the IT folks to create a Shared Data Source using **Report Builder**.

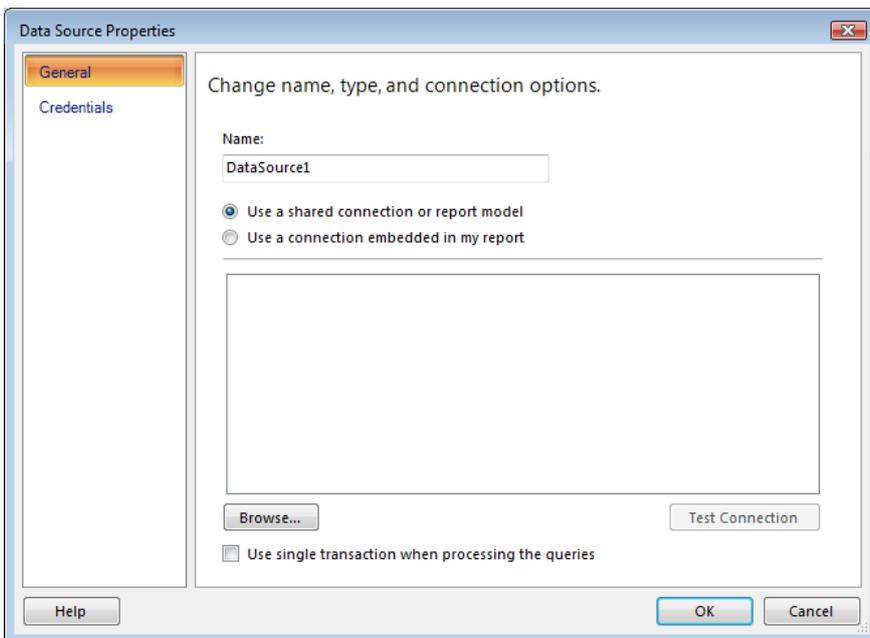
1.3 Data Sources with the Wizard

First we need to create a Data Source for our report.

- Click **Blank Report**.

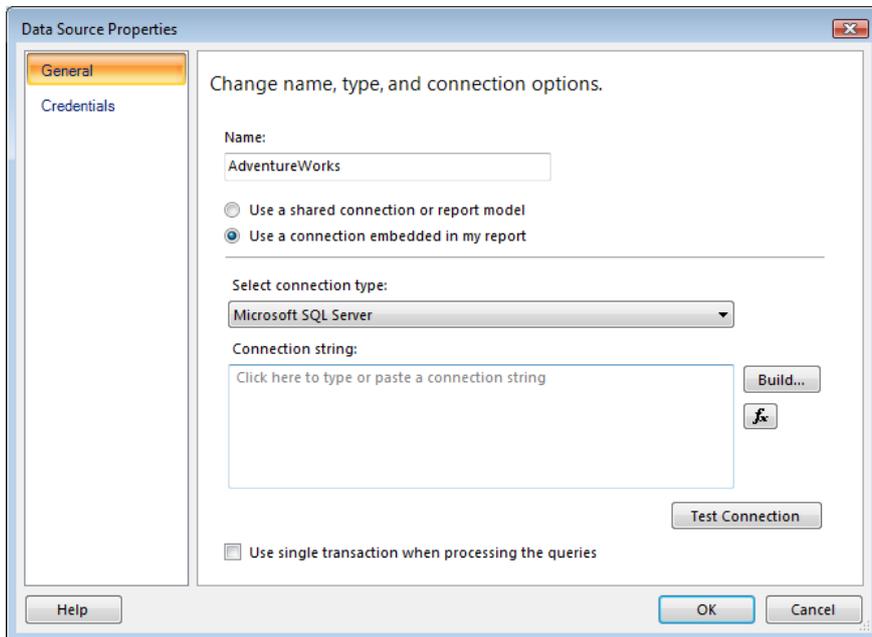


- Right click **Data Sources**.
- Click **Add Data Source ...**

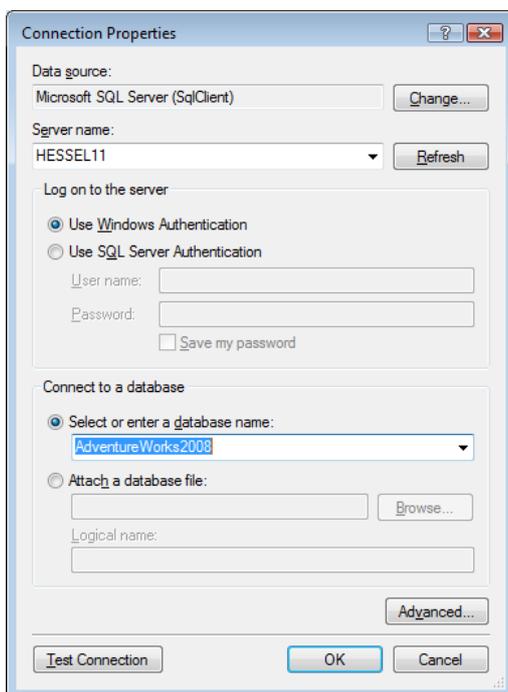


- Fill in **Name: dsrAdventureWorks**.
- Check **Use a connection embedded in my report**.

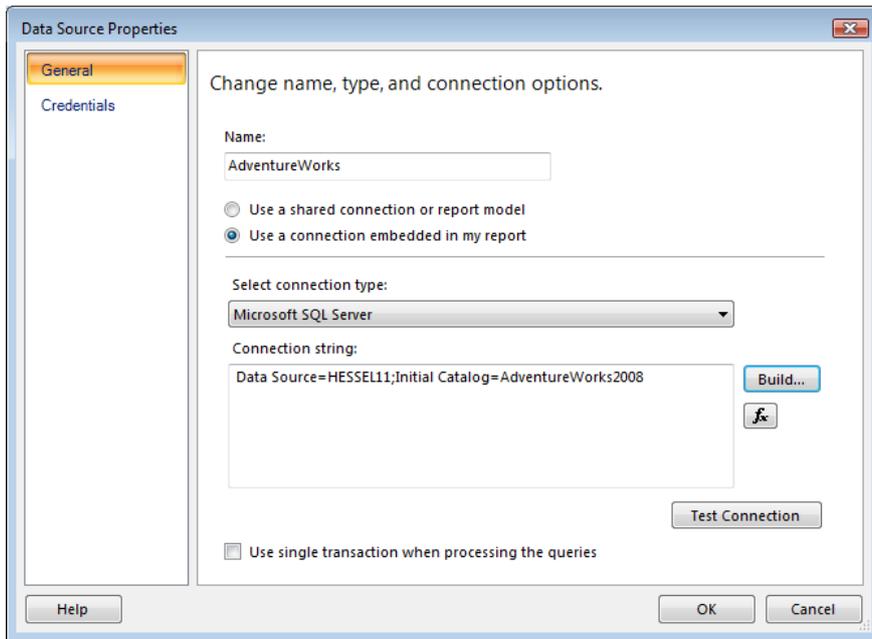
Note: You would use **single transaction** to reduce the amount of open connections to the database. For example, if you have a report with 3 datasets and you don't have this option checked, a new connection is made to the database for every single dataset. However, if you have it checked, then only one connection will be open to the database and all the datasets will return the data and the connection will be closed. This can be used to reduce network traffic and potentially increase performance.



- Click **Build...**

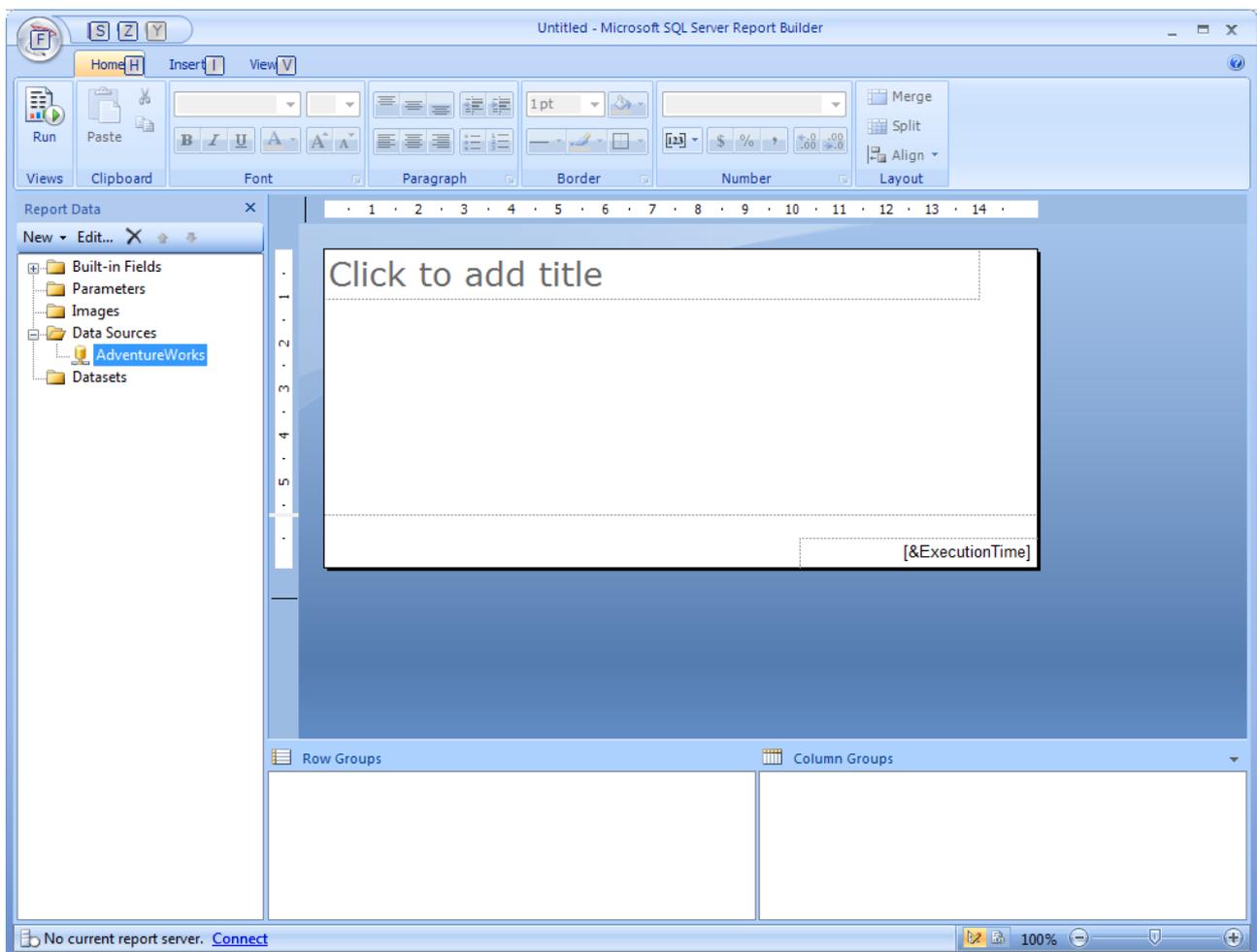


- Select **Server name**.
- Select or enter a database name.
- Click **OK**.



- Click **OK** again.

A new Data Source has been added.

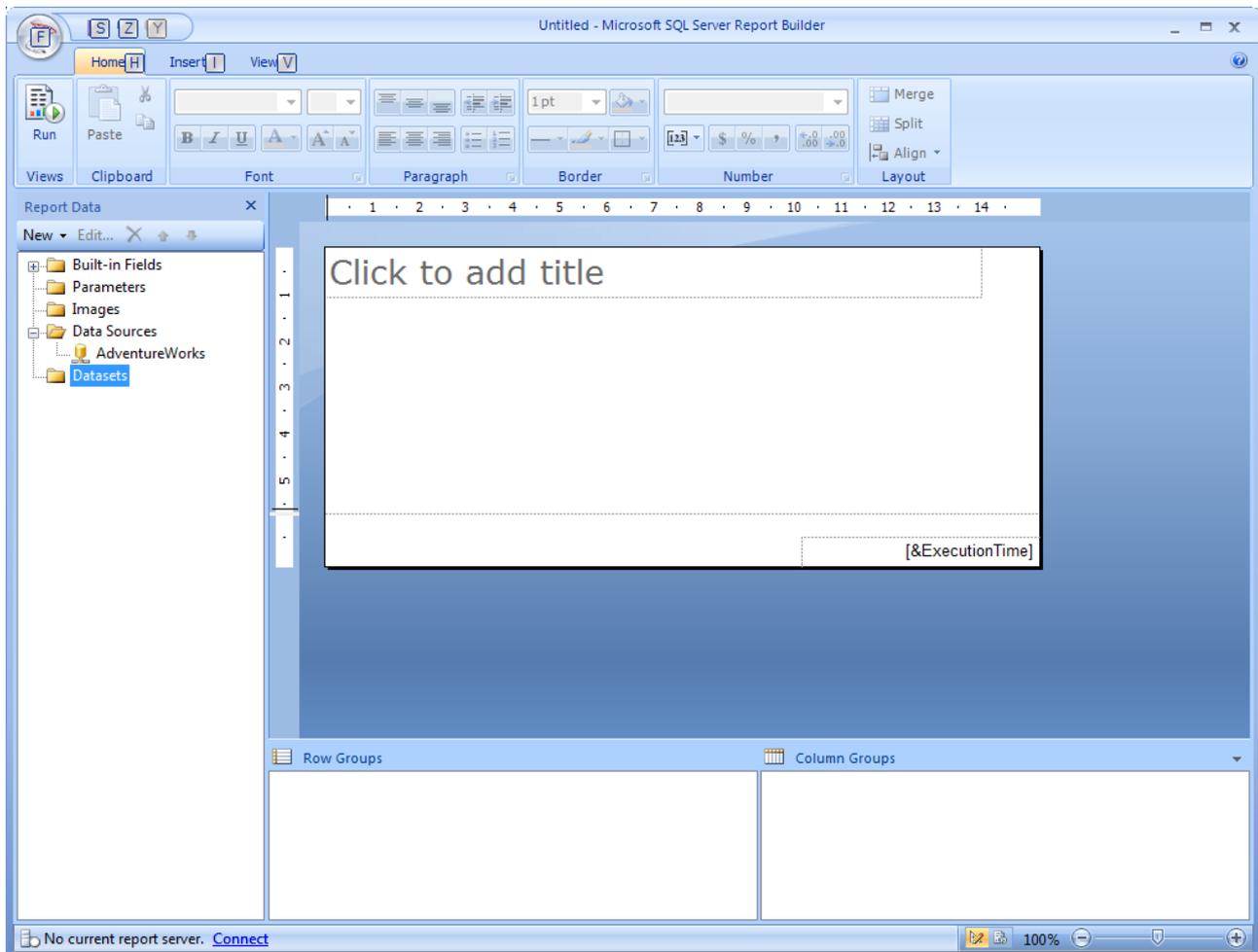


This data source can easily be adjusted by right-clicking it and picking the data source properties.

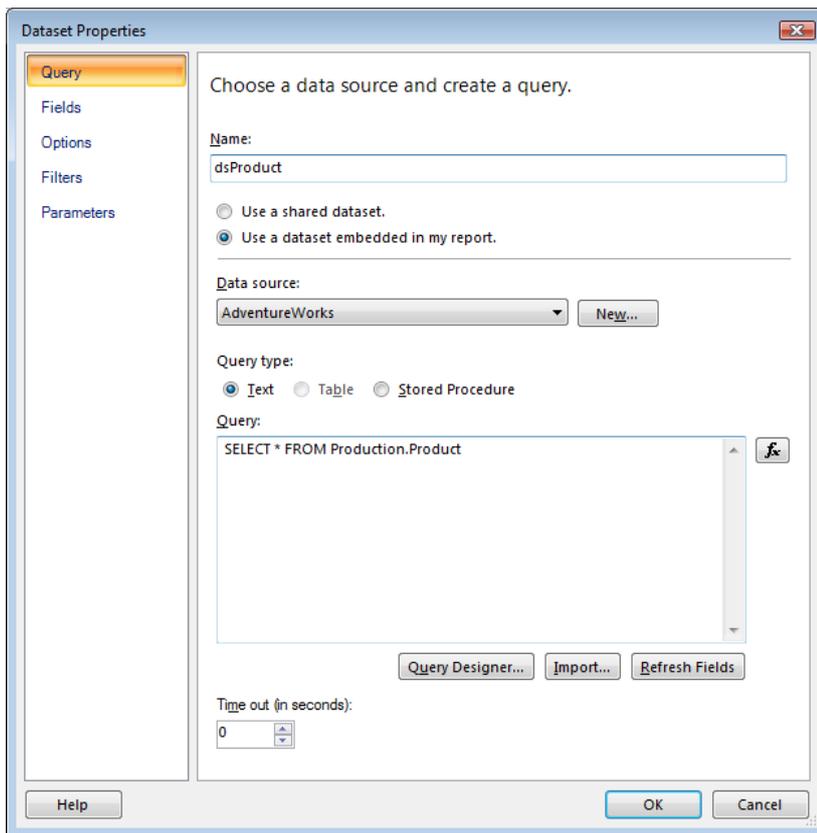
The next step in manually creating a data source is to construct a dataset to be used in the report from the data source. When using the query designer with the wizard as discussed below, a dataset will automatically be created.

1.4 Manual Data Set Construction

Each report uses a query to specify data, the columns and rows that will be used in it. The query designer is an easy to use, drag and drop **GUI** that creates the query statement in the appropriate language, e.g., **SQL** for relational databases, cubes, etc. unless a text-based designer is supported.

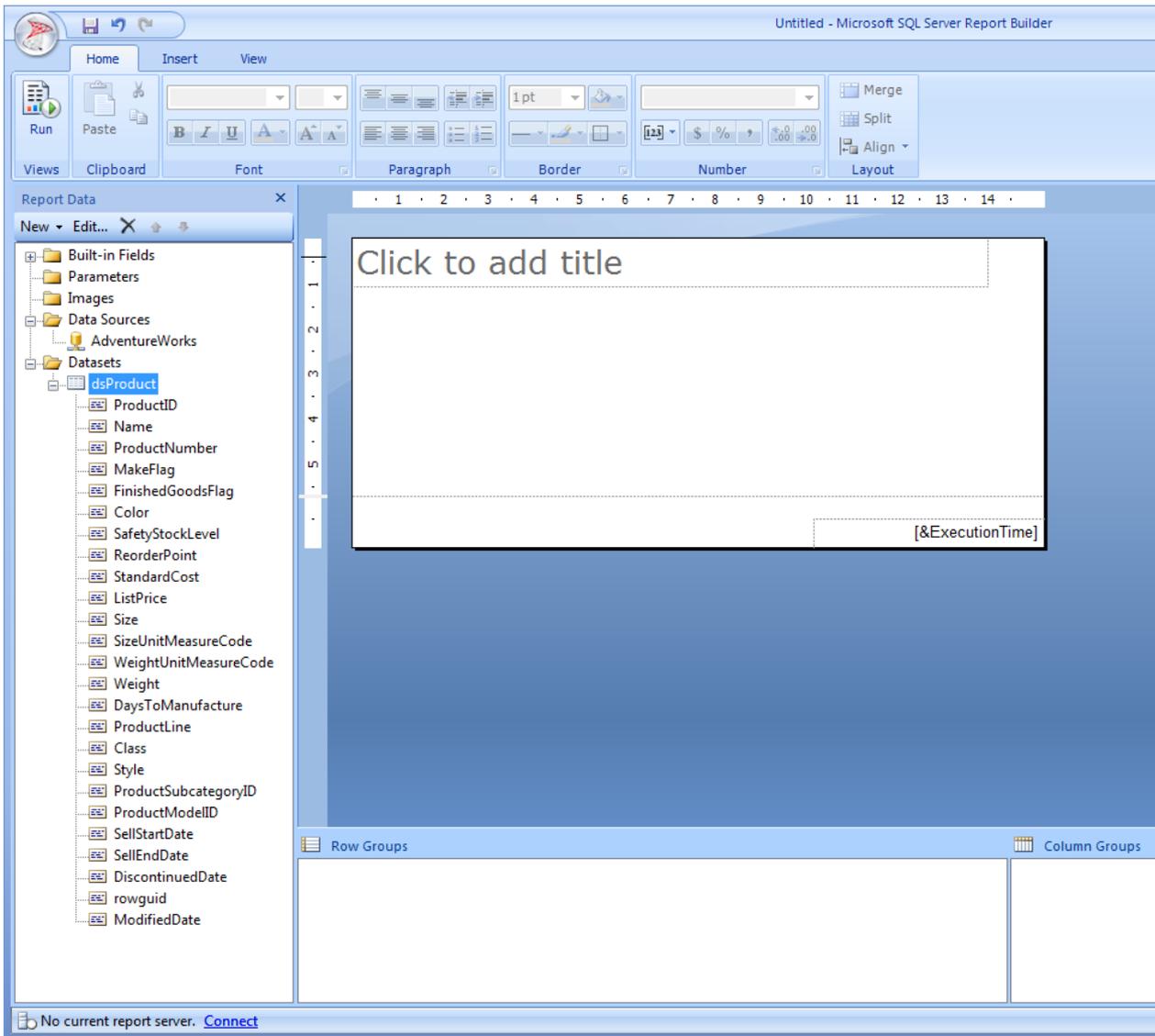


- Right click **Datasets**.
- Click **Add dataset ...**
- Fill it in as shown here:

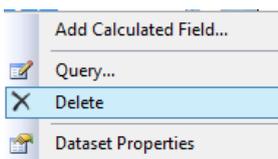


- Click **OK**.

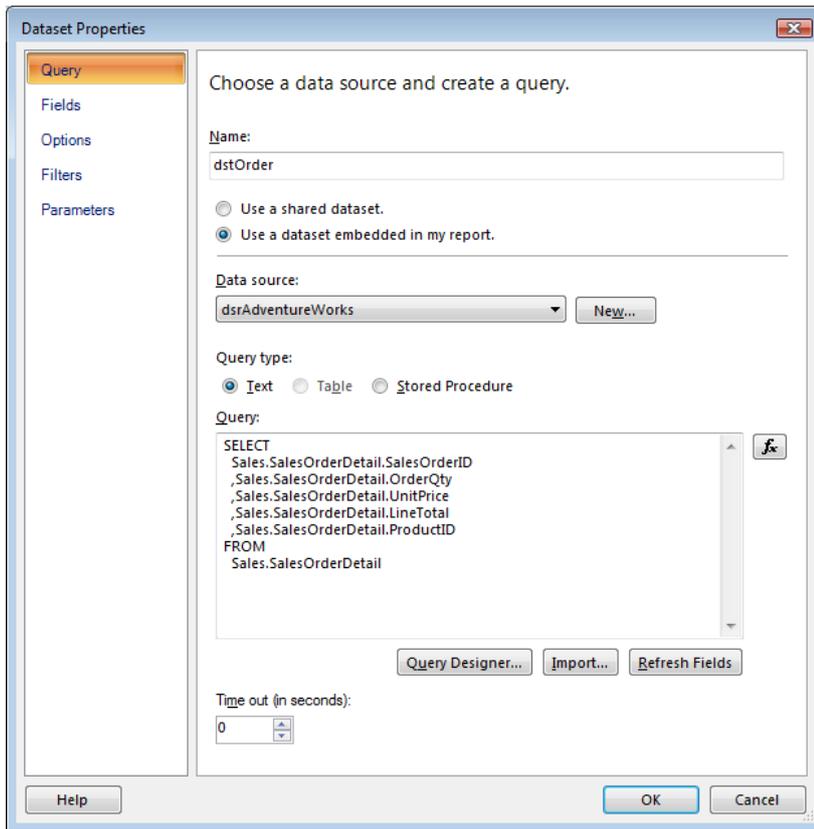
The result should be looking like this:



By right clicking the dataset you get:



By double clicking the dataset you get the **dataset properties**:

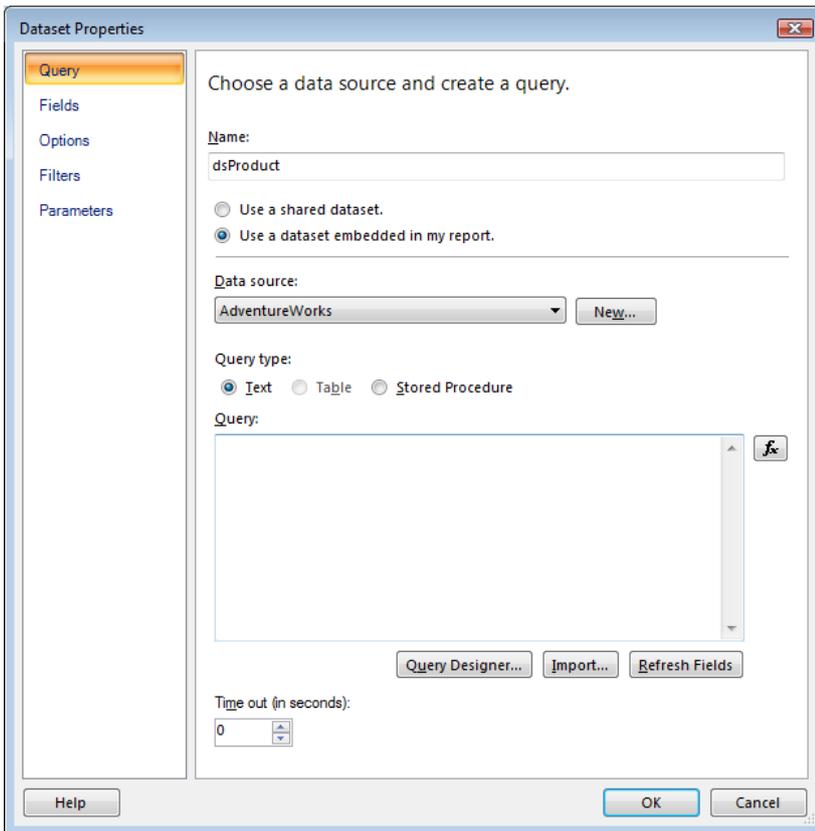


- Save the report as **Report001.rdl**.

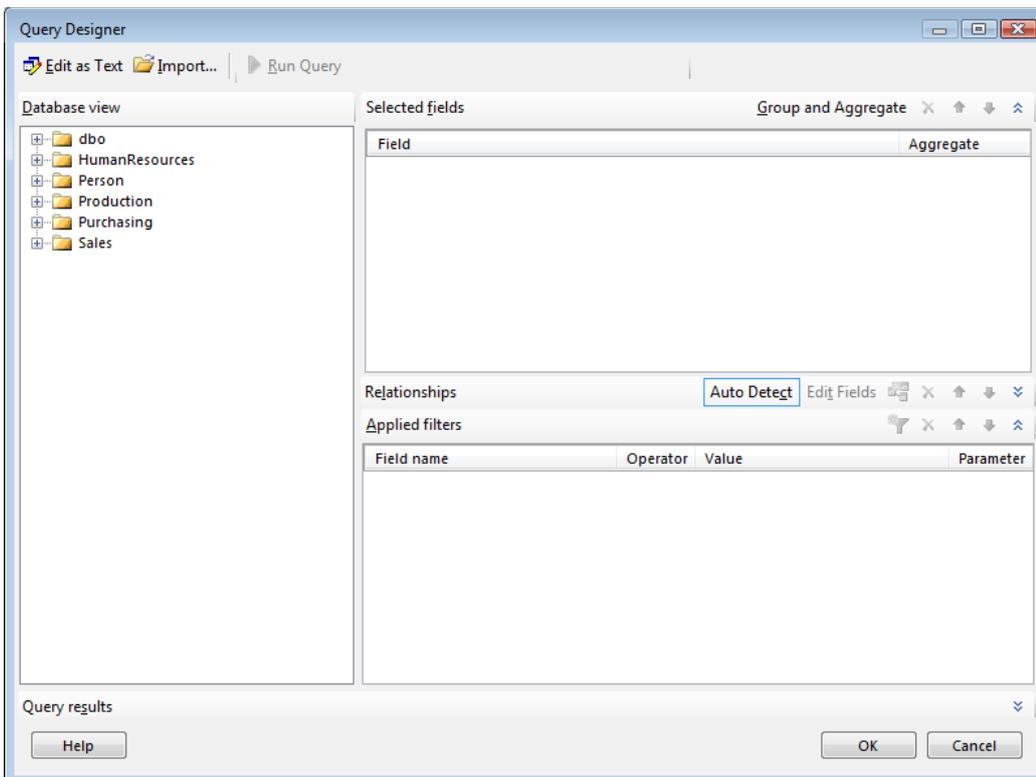
1.5 Data Construction Using The Query Designer

We'll continue with the report we just created, **Report001.rdl**.

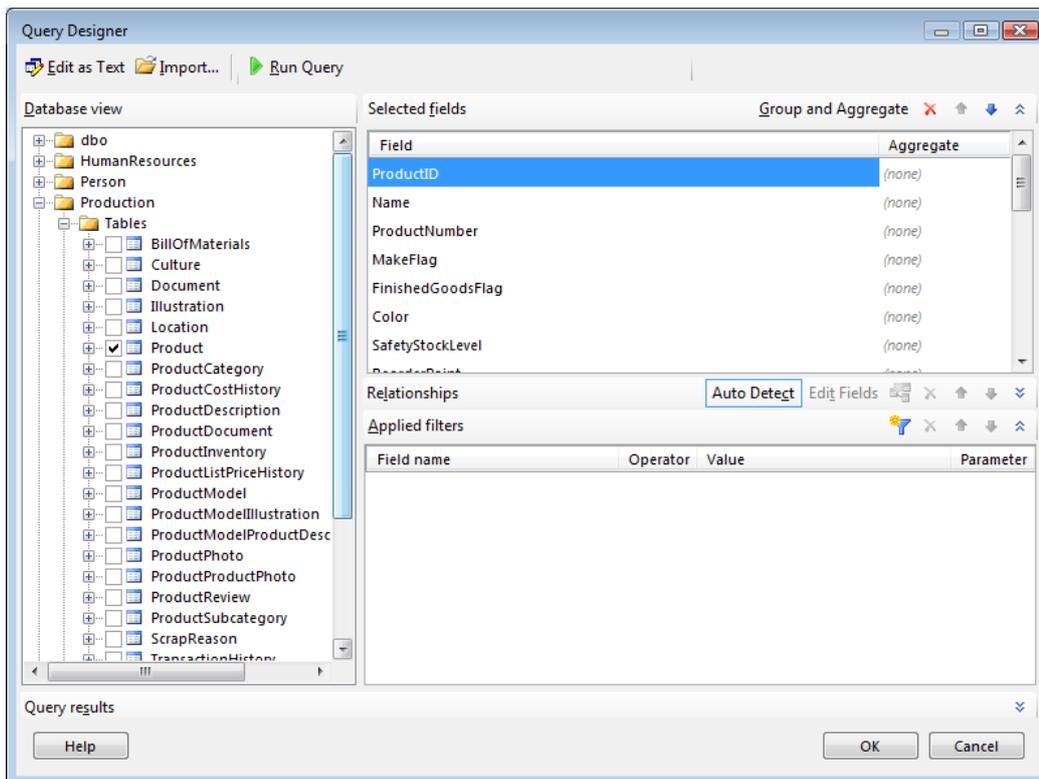
- Remove the existing dataset.
- Right click **Datasets**.
- Click **Add dataset ...**
- Fill it in as shown here:



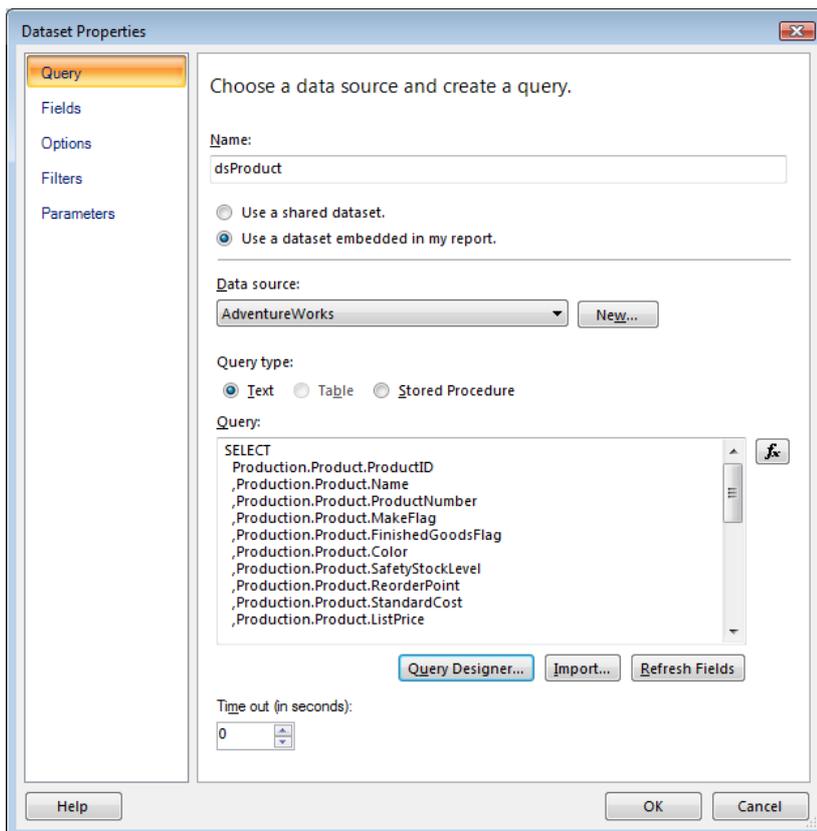
- Click **Query Designer...**



- Click **Production**.
- Click **Tables**.
- Check **Product**.



- Click **OK**.



- Click **OK**.
- Save the report (Report001.rdl).



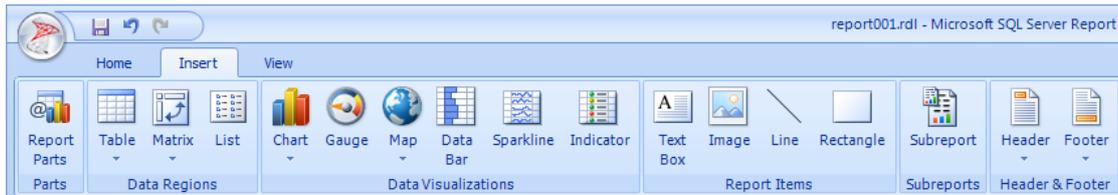
1.6 Data Regions

Data regions are used to display data in a report. There are three types of data regions in Report Builder 3.0:

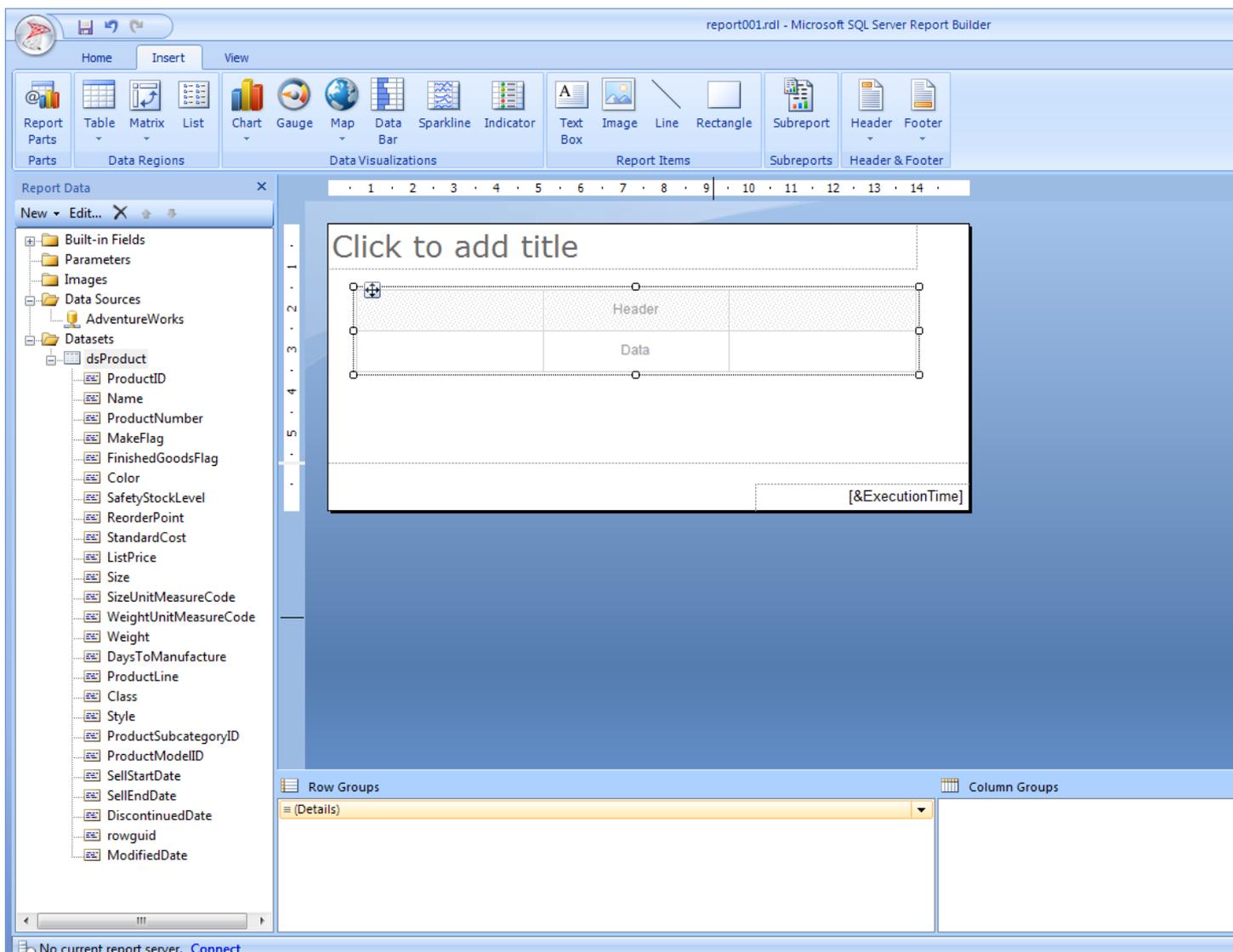
- Tables.
- Matrixes (cross tabs).
- Lists (free placements).

There can be more than one region in a report, and data regions can be nested within other data regions.

- Create a new report.
- Create the data source **dsrcAdventureWorks**.
- Create the dataset **dstProduct**.
- On the ribbon click the **Insert** tab.



- Click Table.
- Click **Insert table**.
- Draw a table in the report.



- Drag the fields **ProductID**, **Name** and **Color** one by one to the table.

Click to add title

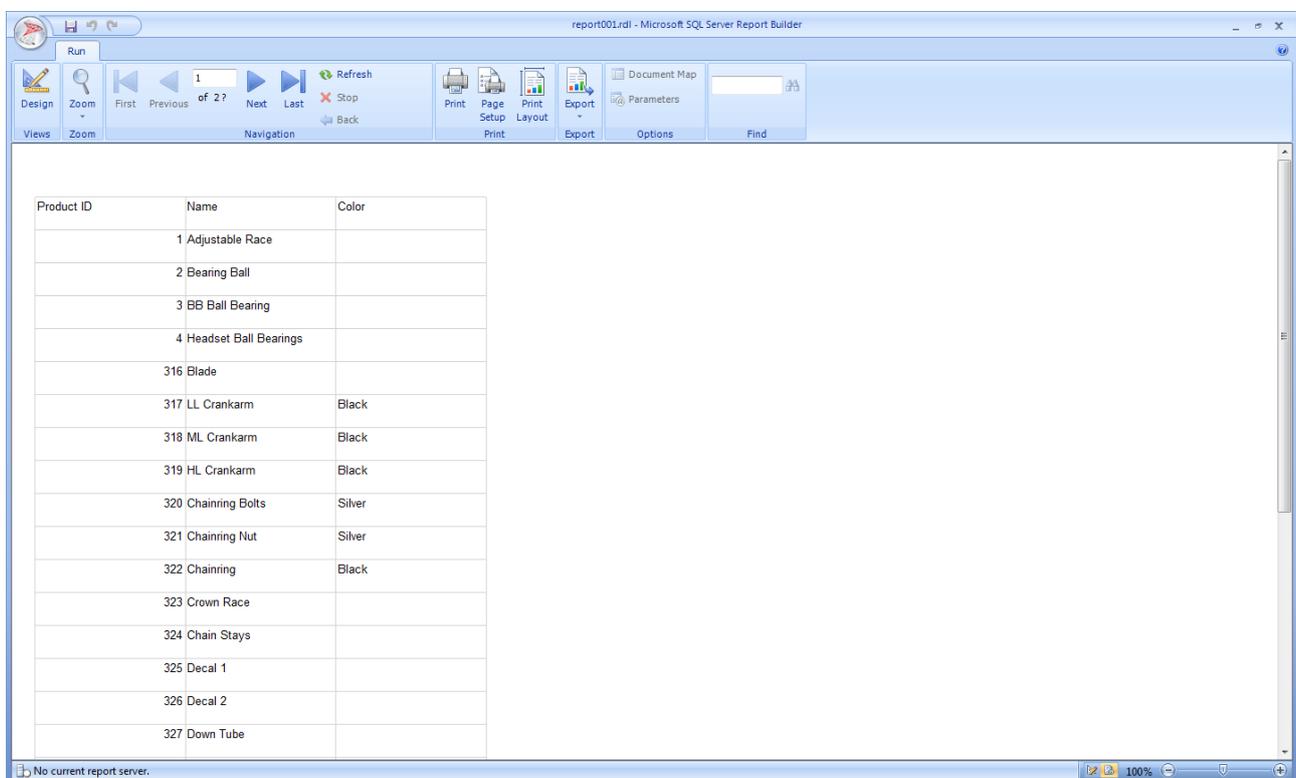
Product ID	Name	Color
[ProductID]	[Name]	[Color]

[&ExecutionTime]

The upper row contains the field names.

- Click the **Home** tab on the ribbon.
- Click **Run**.

It should look like this:

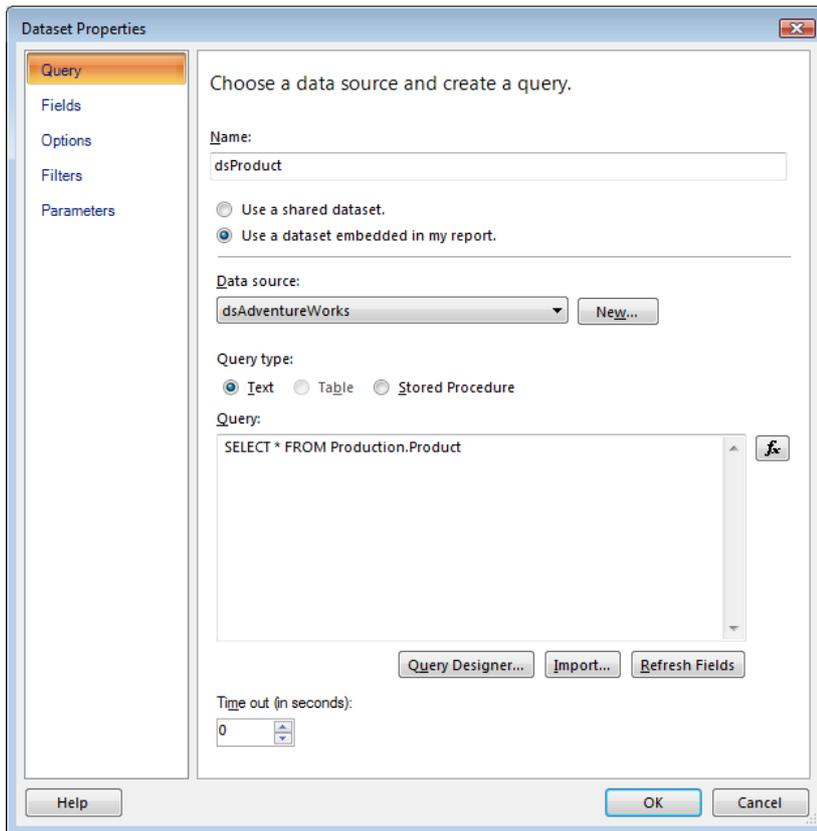


- Click **Design** on the **Home** tab.
- Save the report with **CTRL + S** as **Report002.rdl**.

1.7 More Dataset Properties

- Create a new report.
- Add the data source **dsrAdventureWorks**.
- Add the dataset **dstProduct** by using the manual dataset construction.
- Use as **SQL** instruction:

```
SELECT * FROM Production.Product
```



Add a **Table** to the report for testing purposes.

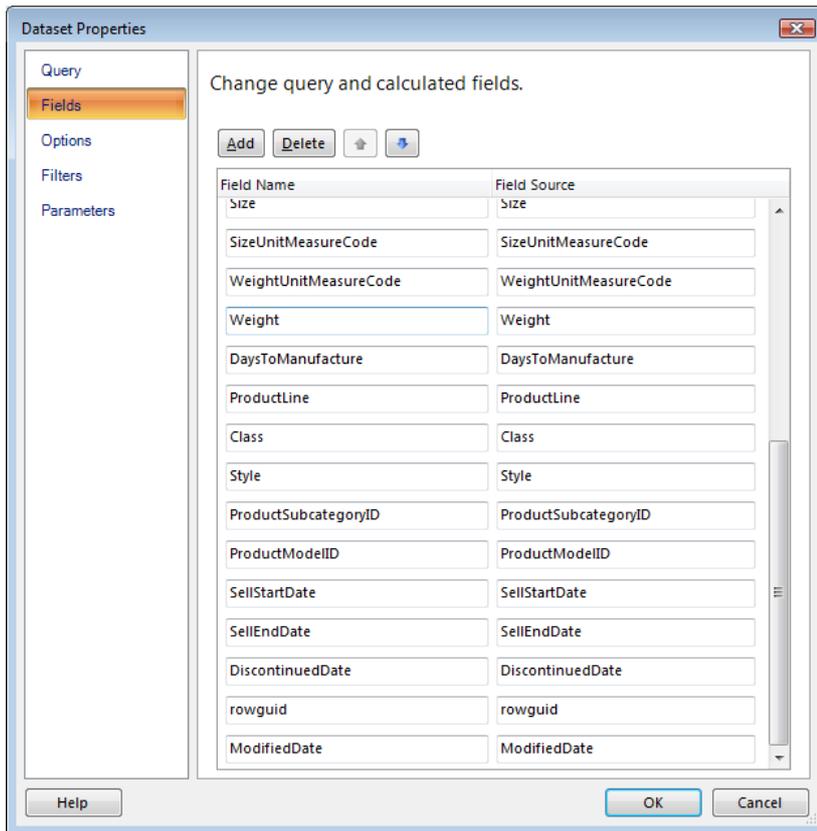
- Click the tab **Insert**.
- Click **Table**.
- Draw a table in your report.

Now we will take a closer look at the other properties:

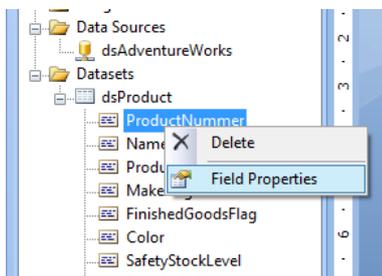
- Fields.
- Options.
- Filters.
- Parameters.

1.7.1 Fields

The pane will look like this:



Tip: we can also get this pane by right clicking one of the fields and choosing **Field Properties**.



In the column **Field Name** we might change our field names and columns.

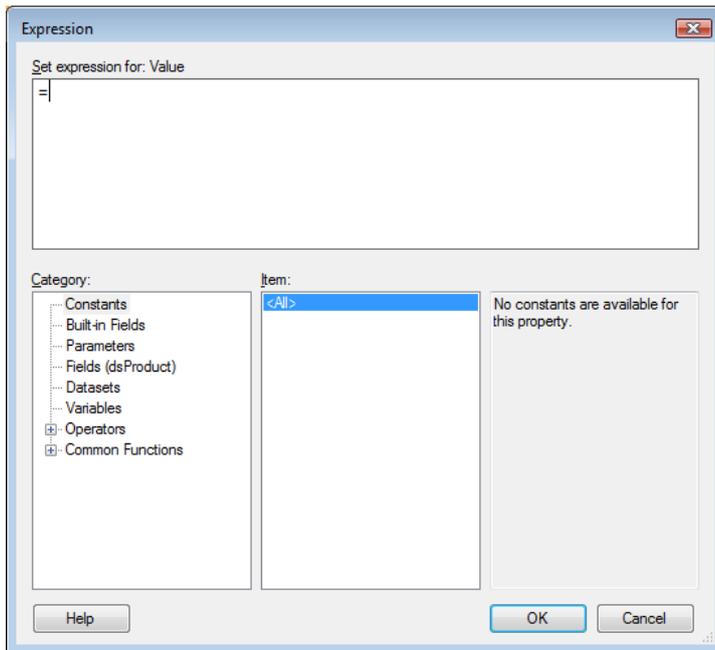
We can also **Add** fields.

- Click **Add**.
- Click **Calculated Field**.

A new field is added at the bottom.

- Call this field **Year**.
- Click 

We'll get this pane:



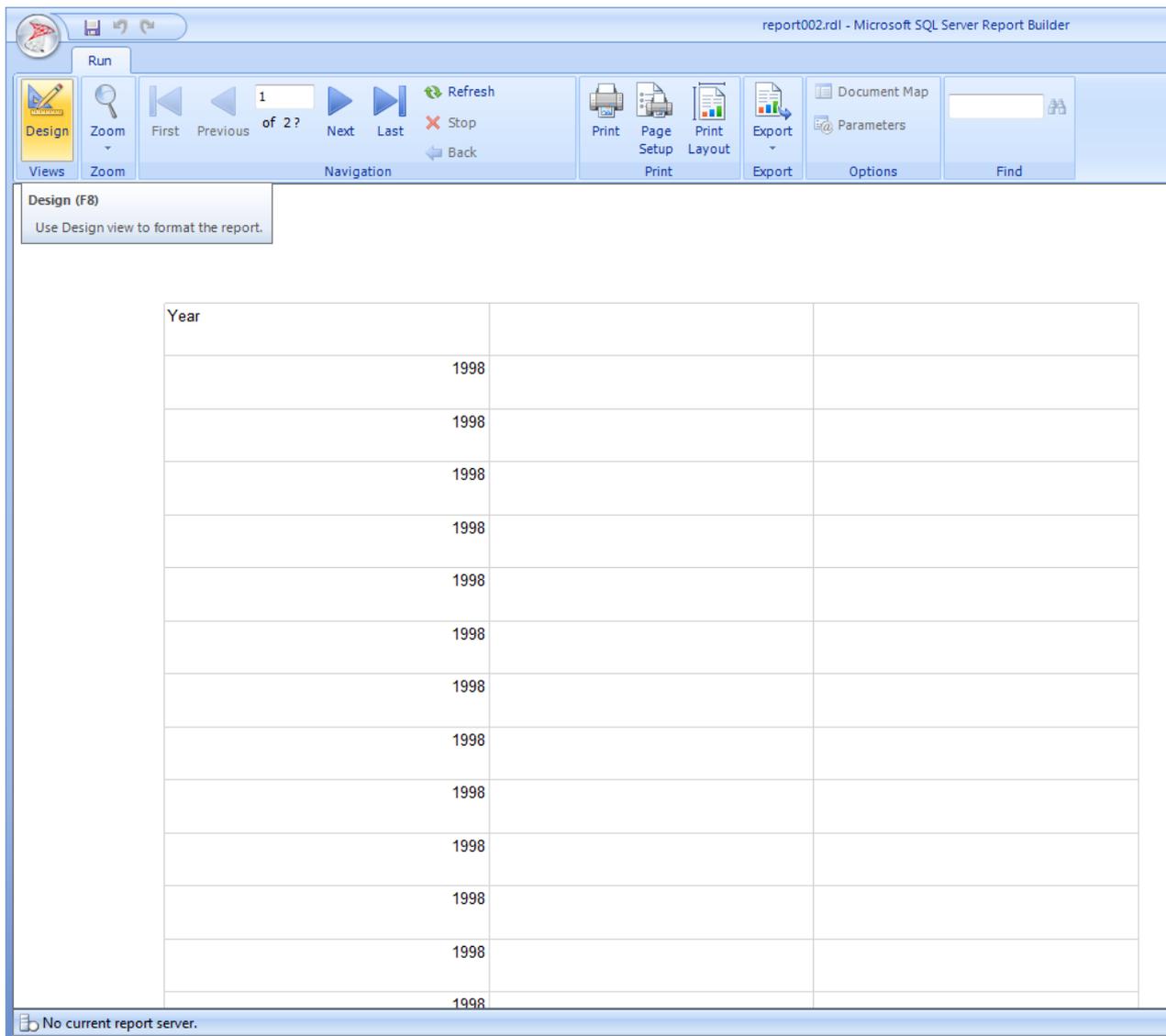
- Click **Common Functions** ⇒ **Date & Time** ⇒ **Year**.
- Click **Fields** ⇒ **SellStartDate.Value**.

The Formula now should look like this:

■ **=Year (Fields!SellStartDate.Value)**

- Click **OK** twice.
- Drag the field **Year** to the first column of the report table.
- On the **Home** tab click **Run**.

We'll get:



Instead of a **Calculated Field** we could also have added a **Query field**.



To add a Query Field:

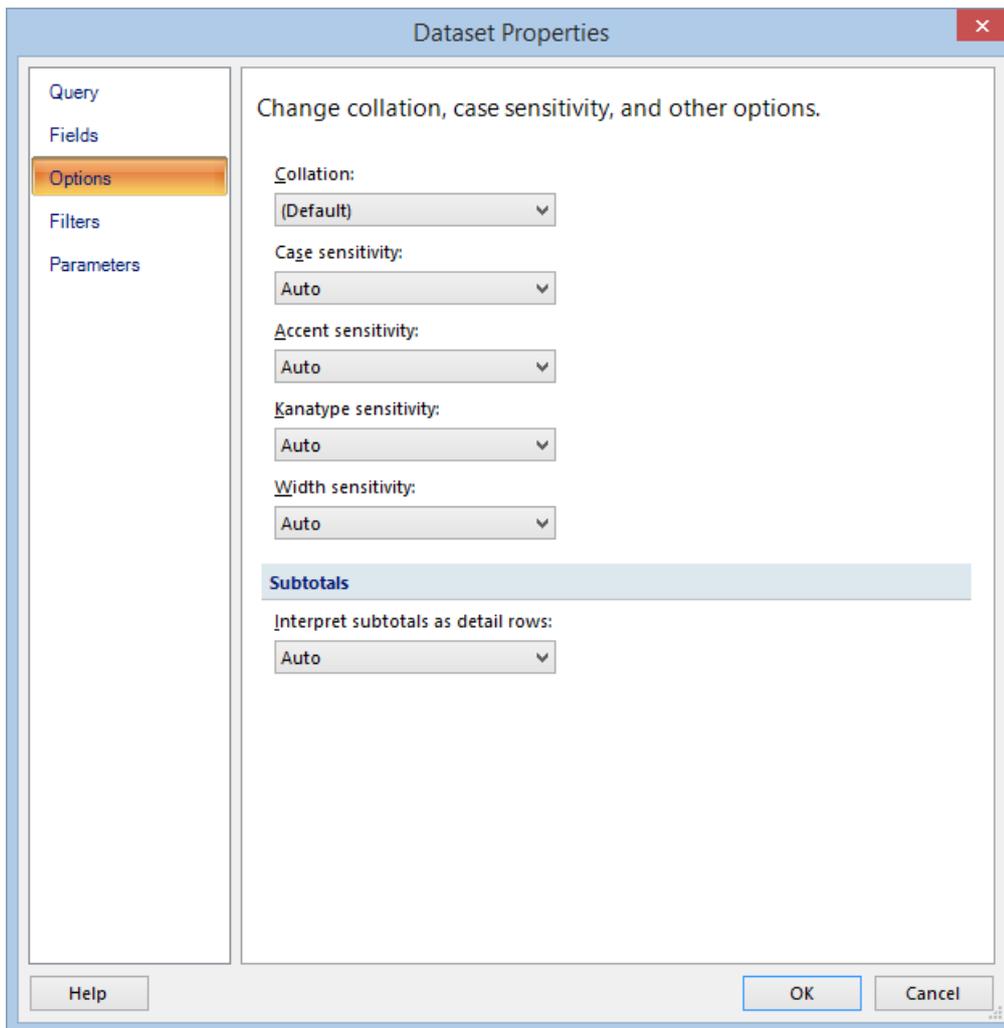
- Click **Query Field**.

A new row is added to the bottom of the grid.

- In the **Field Name** text box, type the name for the field.
- In the **Field Source** text box, type the name of an existing field on the data set.
- Click **OK**.

1.7.2 Options

- In the **Data Set Properties** pane click **Options**.



Collation

Select a locale that determines the collation sequence to be used for sorting data. **Default** indicates that the report server should attempt to derive the value from the data provider when the report runs. If the value cannot be derived, the default value is derived from the locale setting of the computer.

Case sensitivity

Select a value that determines case sensitivity. This option indicates whether the data is case-sensitive. You can set **Case Sensitivity** to **True**, **False**, or **Auto**. The default value, **Auto**, indicates that the report server should attempt to derive the value from the data provider when the report runs. If the data provider does not support the case-sensitivity type, the report runs as though the value were **False**. If you know the value and you know it is supported, choose **True**.

Accent sensitivity

Select a value that determines accent sensitivity. **Accent Sensitivity** indicates whether the data is accent sensitive and can be set to **True**, **False**, or **Auto**. The default value, **Auto**, indicates that the report server should attempt to derive the value from the data provider when the report is run. If the data provider does not support the accent sensitivity type, the report runs as though the value were **False**. If you know the value and you know it is supported, choose **True**.

Kanatype sensitivity

Select a value that determines kanatype sensitivity. This option indicates whether the data is kanatype sensitive; it can be set to **True**, **False**, or **Auto**. The default value, **Auto**, indicates that the report server should attempt to derive the value from the data provider when the report runs. If the data provider does not support the kanatype sensitivity type, the report runs as though the value were **False**. If you know the value and you know it is supported, choose **True**.

Width sensitivity

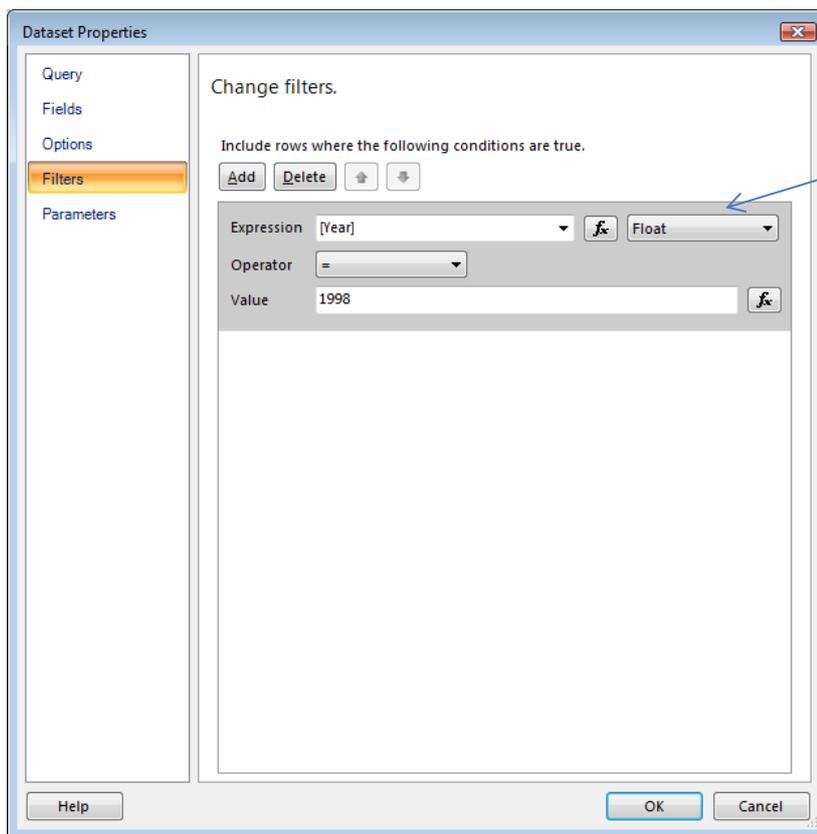
Select a value that determines width sensitivity. This option indicates whether the data is width-sensitive and can be set to **True**, **False**, or **Auto**. The default value, **Auto**, indicates that the report server should attempt to derive the value from the data provider when the report runs. If the data provider does not support the width sensitivity type, the report runs as though the value were **False**. If you know the value and you know it is supported, choose **True**.

Interpret subtotals as detail rows

Select a value that indicates whether you want subtotal rows to be interpreted as detail rows instead of aggregate rows. The default value, **Auto**, indicates that the subtotal rows should be treated as detail rows if the report does not use the **Aggregate()** function to access any fields in the data set. If you want subtotal rows to be interpreted as aggregate rows, choose **False**. If you want the subtotal rows to be interpreted as detail rows and you know that they do not use the **Aggregate()** function, choose **True**.

1.7.3 Filters

- In the **Data Set Properties** screen click **Filters**.



- Click **Add**.
- Fill it in as shown in the illustration above (Note: The expression must have the right data type!).
- Click **OK**.
- Test the outcome.
- Save the report as **Report003.rdl**.

1.7.4 Complex formulas

Just some examples.

To calculate a persons age:

```
=iif(month(Fields!BirthDate.Value)<month(Today()),
year(today()-year(Fields!BirthDate.Value)-1,
iif(
month(Fields!BirthDate.Value)>month(Today()),
year(today()-year(Fields!BirthDate.Value),
iif(
day(Fields!BirthDate.Value)<day(Today()),
year(today()-year(Fields!BirthDate.Value),
year(today()-year(Fields!BirthDate.Value)-1
)
)
)
```

Obviously, this can also be done in **SQL** with a **CASE WHEN** construction.

Dutch week numbers

```
=DatePart("ww",Fields!SellStartDate.Value,FirstDayOfWeek.Monday,
FirstWeekOfYear.FirstFourDays)
```

Concatenating text fields

```
=Fields!FirstName.Value & vbCrLf & Fields!LastName.Value
```

Switch example

```
=switch
(
year(Fields!SellStartDate.Value)=1996,"A",
year(Fields!SellStartDate.Value)=1997,"B",
year(Fields!SellStartDate.Value)=1998,"C"
)
```

1.7.5 Parameters

Query Parameter

When you add a query parameter to a query, **Report Builder** automatically creates a single-valued report parameter with default properties for name, prompt, and data type.

- Create a new report.
- Save the report as **Report004.rdl**.
- Add the data source **dssAdventureWorks**.
- Add the dataset **dstProduct** by using the manual dataset construction.
- Use **SQL** instruction:

```
SELECT * FROM Production.Product
```

- Add the following **Transact-SQL WHERE** clause as the last line in the query:

```
WHERE ProductID = (@ProductID)
```

As we can see, a parameter is automatically added, at the left in **Parameters**.

The **WHERE** clause limits the retrieved data to the store identifier that is specified by the query parameter **@ProductID**.

- Add a table to the report.
- Add the fields **ProductID**, **Name**, **Color** to the table.
- Save the report as **Report005.rdl**.
- On the query designer toolbar, click **Run (!)**.
-

The **Define Query Parameters** dialog box opens and prompts for a value for the query parameter **@ProductID**.

- In **Parameter Value**, type **320**.
- Click **View Report**.
- Click **Design**.
- In the Report Data pane, expand the **Parameters** folder.

Notice that there is now a report parameter named **@ProductID**. By default, the parameter has the data type **Text**. Because the store identifier is an **Integer**, you will have to change the data type to **Integer** in the next procedure. After a report parameter is created, you can adjust the default values for properties.

To change the default data type for a report parameter

- In the Report Data pane under the **Parameters** node, right-click **@ProductID**.
- Click **Parameter Properties**.
- In **Prompt**, type **Product identifier?**

This text appears on the report viewer toolbar when you run the report.

- In **Data type**, from the drop-down list, select **Integer**.
- Accept the remaining default values in the dialog box.
- Click **OK**.
- Preview the report.

The report viewer displays the prompt for **@ProductID**.

- On the report viewer toolbar, next to **Product identifier**, type **320**, and then click **View Report**.

To ensure a user can only type valid values for a parameter, you can create a drop-down list of values to choose from. The values can come from a dataset or from a list you specify. Available values must be supplied from a dataset that has a query that does not contain a reference to the parameter.

To create a dataset for valid values for a parameter

- Switch to Design view.
- In the Report Data pane, right-click the **Datasets** folder.
- Click **Add Dataset**.
- In **Name**, type **dstProductParameter**.
- Select the **Use a dataset embedded in my report** option.
- In **Data source**, from the drop-down list, choose the data source you created in the first procedure.
- In **Query type**, verify that **Text** is selected.
- In **Query**, paste the following text:

```
SELECT
Production.Product.ProductID
,Production.Product.Name
FROM
Production.Product
ORDER BY Production.Product.Name
```

- Click **OK**.

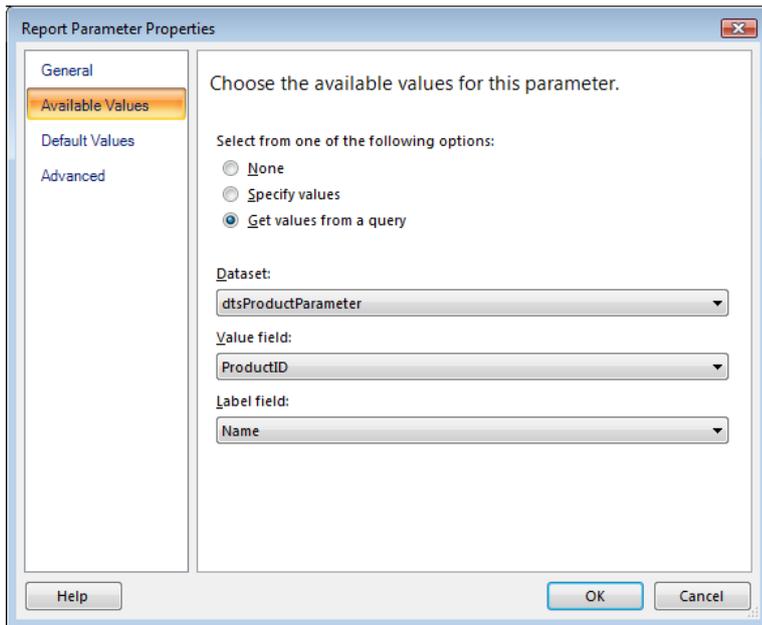
The Report Data pane displays the fields **ProductID** and **Name** below the **dstProductParameter** dataset node.

Specify Available Values to Create a Drop-down List of Values

After you create a dataset to provide available values, you must change the report properties to specify which dataset and which field to use to populate the drop-down list of valid values on the reportviewer toolbar.

To provide available values for a parameter from a dataset:

- In the Report Data pane, right-click the parameter **@ProductID**.
- Click **Parameter Properties**.
- Click **Available Values**.
- Click **Get values from a query**.



- In **Dataset**, from the drop-down list, click **dtsProductParameter**.
- In **Value field**, from the drop-down list, click **ProductID**.
- In **Label field**, from the drop-down list, click **Name**.

The label field specifies the display name for the value.

- Click **General**.
- In Prompt, type **Product name?**

The user will now select from a list of store names instead of store identifiers. Note that the parameter data type remains **Integer** because the parameter is based on the store identifier, not the store name.

- Click **OK**.
- Preview the report.

In the report viewer toolbar, the parameter text box is now a drop-down list that displays **<Select a Value>**.

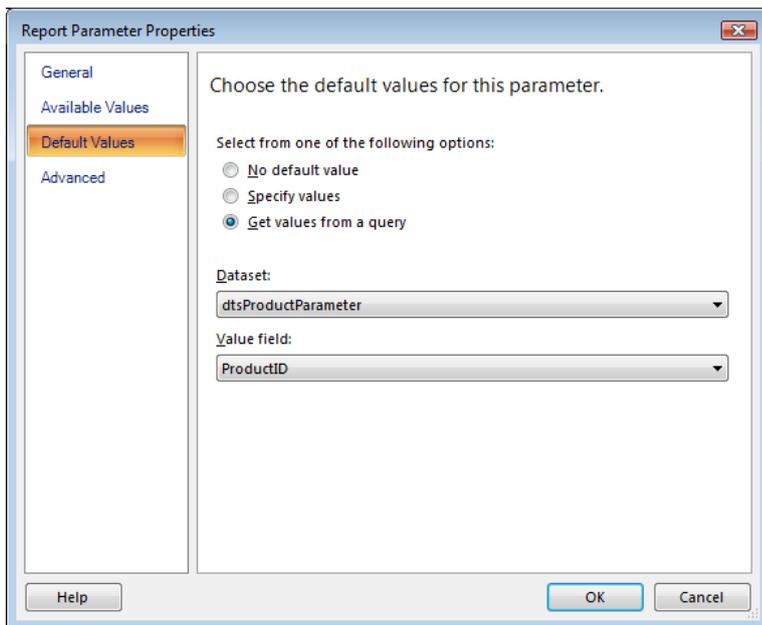
- From the drop-down list, select **Cone-Shaped Race**.
- Then click **View Report**.

Specify Default Values so the Report Runs Automatically

You can specify a default value for each parameter so the report runs automatically.

To specify a default value from a dataset.

- Switch to Design view.
- In the Report Data pane, right-click **@ProductID**, and then click **Parameter Properties**.
- Click **Default Values**.
- Click **Get values from a query**.
- In **Dataset**, from the drop-down list, click **dtsProductParameter**.



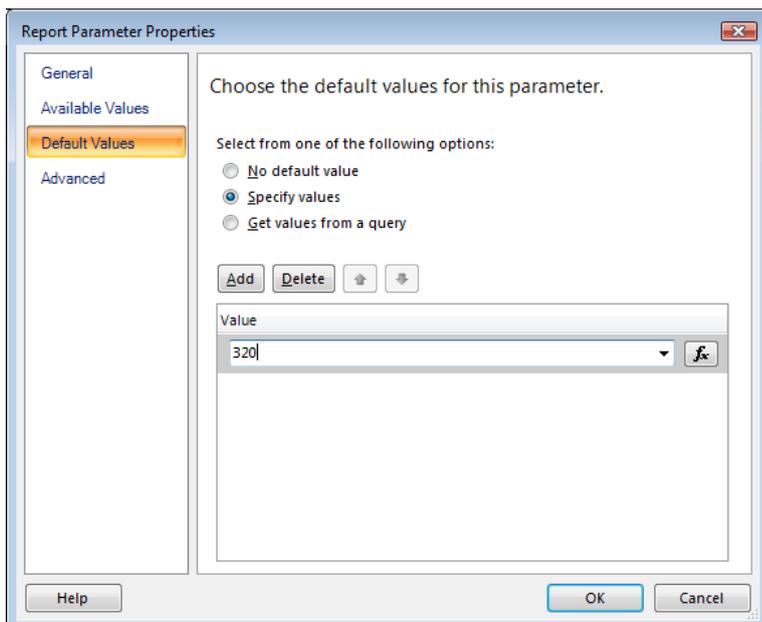
- In **Value field**, from the drop-down list, click **ProductID**.
- Click **OK**.
- Preview the report.

For **@ProductID**, the report viewer displays the value *Adjustable Race*. This is the first value from the result set for the dataset **dtsProductParameter**.

To specify a custom default value

- Switch to Design view.
- In the Report Data pane, right-click **@ProductID**.
- Click **Parameter Properties**.
- Click **Default Values**.
- Click **Specify values**.
- Click **Add**.

A new value row is added.



- In **Value**, type **320**.
- Click **OK**.
- Preview the report.

Look up a Value from a Dataset that has Name/Value Pairs

A dataset might contain both the identifier and the corresponding name field. When you only have an identifier, you can look up the corresponding name in a dataset that you created that includes name/value pairs.

To look up a value from a dataset

- Switch to Design view.
- On the design surface, in the matrix, in the first row column header, right-click **[ProductID]**
- Click **Expression**.
- In the expression pane, delete all text except the beginning **equals (=)**.
- In **Category**, expand **Common Functions**, and click **Miscellaneous**.

The Item pane displays a set of functions.

- In Item, double-click **Lookup**.

The expression pane displays

```
=Lookup (
```

The Example pane displays an example of Lookup syntax.

- Type the following expression:

```
=Lookup (Fields!ProductID.Value, Fields!ProductID.Value,  
Fields!Name.Value, "dstProduct")
```

The **Lookup** function takes the value for **ProductID**, looks it up in the **dstProduct**" dataset, and returns the **Name** value.

- Click **OK**.

The store column header contains the display text for a complex expression: <<**Expr**>>.

- Preview the report.

The text box at the top of each page displays the product name instead of the product identifier.

Display the Selected Parameter Value in the Report

When a user asks questions about a report, it helps to know which parameter values they chose. You can preserve user-selected values for each parameter in the report. One way is to display the parameters in a text box in the page footer.

To display the selected parameter value and label on a page footer

- Switch to Design view.
- Right-click the page footer.
- Point to **Insert**.
- Click **Text Box**.
- Drag the text box next to the text box with the time stamp.
- Grab the side handle of the text box and expand the width.
- From the Report Data pane, drag the parameter **@ProductID** to the text box.

The text box displays

```
[@ProductID]
```

- To display the parameter label, click in the text box until the insert cursor appears after the existing expression.
- Type a space.
- Then drag another copy of the parameter from the Report Data pane to the text box.

The text box displays:

```
■ [ @ProductID ] [ @ProductID ]
```

- Right-click the first expression
- Click **Expression**.

The **Expression** dialog box opens.

- Replace the text **Value** by **Label**.
- Click **OK**.

The text displays:

```
■ [ @ProductID.Label ] [ @ProductID ]
```

- Preview the report.

Use the Report Parameter in a Filter

Filters help control which data to use in a report after it is retrieved from an external data source. To let a user help control the data they want to see, you can include the report parameter in a filter for the matrix.

To specify a parameter in a matrix filter

- Switch to Design view.
- Right-click a row or column header handle on the matrix
- Then click **Tablix Properties**.
- Click **Filters**.
- Click **Add**.

A new filter row appears.

- In **Expression**, from the drop-down list, select the dataset field **ProductID**.

The data type displays **Integer**. When the expression value is a dataset field, the data type is set automatically.

- In **Operator**, verify that **equals (=)** is selected.
- In **Value**, type

```
■ [ @ProductID ] .
```

[@ProductID] is the simple expression syntax that represents

```
■ =Parameters!ProductID.Value
```

If we choose to do so by using the mouse, we get:

```
■ =Parameters!ProductID.Value (0)
```

Which is not working! So, we have to remove **(0)** by hand.

- Click **OK**.
- Preview the report.

The matrix displays data only for **Chainrings Bolts**.

- On the report viewer toolbar, for **Product name?**, select **Chainrings Nut**.
- Then click **View Report**.
- Save the report (**Report005.rdl**).

The matrix displays data corresponding to the store you selected.

Change the Report Parameter to Accept Multiple Values

To change a parameter from **single** to **multivalued**, you must change the query, and all expressions that contain a reference to the parameter, including filters. A multivalued parameter is an array of values. In a dataset query, query syntax must test for inclusion of one value in a set of values. In a report expression, expression syntax must access an array of values instead of an individual value.

To change a parameter from single to multivalued

- In the Report Data pane, right-click **@ProductID**.
- Then click **Parameter Properties**.
- Select **Allow multiple values**.
- Click **OK**.
- In the Report Data pane, expand the **Datasets** folder.
- Right-click **dstProduct**, and then click **Query**.
- Change **equals (=)** to **IN** in the **WHERE** clause in the last line in the query:

```
WHERE StoreID IN (@ProductID)
```

The **IN** operator tests a value for inclusion in a set of values.

- Click **OK**.
- Right-click a row or column header handle on the matrix, and then click **Tablix Properties**.
- Click **Filters**.
- In **Operator**, select **In**.
- Click **OK**.

- In the text box that displays the parameter in the page footer, delete all text.
- Right-click the text box.
- Then click **Expression**.
- Type the following expression:

```
=Join(Parameters!StoreID.Label, ", ")
```

The multivalued parameter is now an array; the function **Join** concatenates all store names that the user selected.

- Click **OK**.
- Click in the text box in front of the expression that you just created.
- Then type the following: **Parameter Values Selected:**
- Preview the report.
- Click the drop-down list next to **Product Name?**

Each valid value appears next to a check box.

- Click **Select All**.
- Then click **View Report**.
- Save the report (**Report005.rdl**).

1.7.6 Expressions that refer to multivalued parameters

When you refer to a parameter in an expression, you use the built-in **Parameters** collection. When using multivalued parameters in expressions, you need to understand both how to address a single value and the entire array of values. The following table provides examples and descriptions of parameter properties for parameters that are set to allow multiple values.

Example	Description
Parameters!<ParameterName>.Value	An array of variant data values for the parameter.
Parameters!<ParameterName>.Label	An array of strings that are labels for the parameter.

Parameters!<ParameterName>.IsMultiValue	Boolean property indicating whether the parameter Allow multiple values option has been selected.
Parameters!<ParameterName>.Count	The number of values in the array.
Parameters!<ParameterName>.Value(0)	The first value in a multivalue array.
Parameters!<ParameterName>.Label(0)	The first label in a multivalue array.
Parameters!<ParameterName>.Value(Parameters! <ParameterName>.Count-1)	The last value in a multivalue array.
Parameters!<ParameterName>.Label(Parameters! <ParameterName>.Count-1)	The last label in a multivalue array.
=Join(Parameters!<ParameterName>.Value, ", ")	An expression that concatenates all the values in the array of a multivalue parameter of type String into one string.
=Split("Value1, Value2, Value3", ",")	Takes a string and creates an array of objects that can be used to pass to a subreport or drillthrough report expecting a multivalue parameter.

You can use **SPLIT** and **JOIN** functions to separate or combine values in the array in any expression. You can use **STRING** and **CINT** functions to convert the values into strings or integers.

1.7.7 Parameter and Ranges

When using date parameters, you may want to narrow a search down to a specific time.

- Create two time parameters as a Data Type.

Below are two examples of how the data for time may be entered.

Start Date	<input type="text"/>	Start Time	00:00
End Date	<input type="text"/>	End Time	23:59

Here is an example of the code using CASE to determine how to build out the SQL statement:

```

SELECT * FROM transaction_table
WHERE transaction_date >=
CASE WHEN substring(@StartTime, 3,1) = ':' THEN
    @StartDate + CAST(@StartTime as smalldatetime)
ELSE
    @StartDate + cast(substring(@StartTime, 1,2) + ':' +
    substring(@StartTime, 3,2) as smalldatetime)
END
AND transaction_date <=
CASE WHEN substring(@EndTime, 3,1) = ':' THEN
    @EndDate + cast(@EndTime as smalldatetime)
ELSE
    @EndDate + cast(substring(@EndTime, 1,2) + ':' + substring(@EndTime,
    3,2) as smalldatetime)
END
END

```

1.8 Sorting the data

1.8.1 Apply Sorting to the Column Headers.

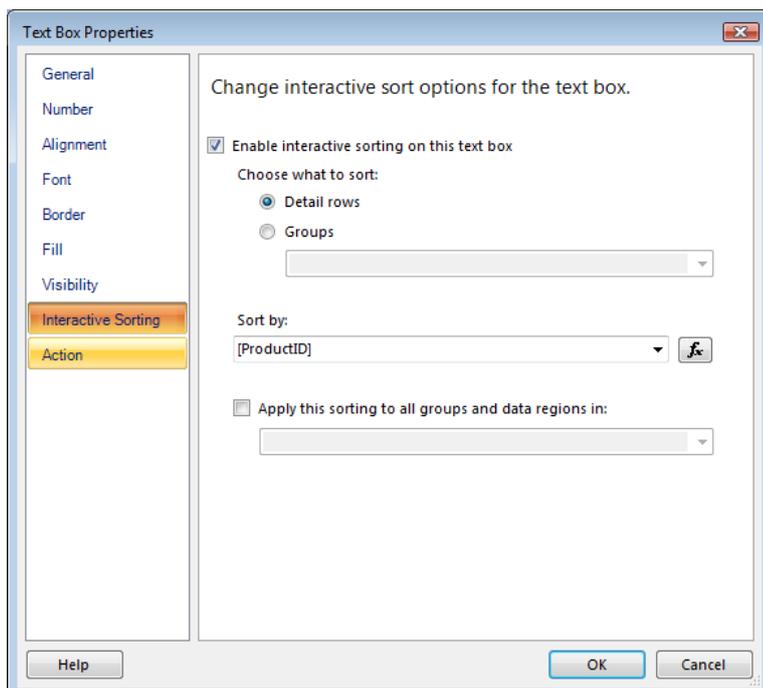
- Create a new report.
- Add the data source **dswAdventureWorks**.
- Add the dataset **dstProduct** by using the manual dataset construction.
- Use as **SQL** instruction:

```

SELECT
Production.Product.ProductID
,Production.Product.Name
,Production.Product.Color
FROM
Production.Product
WHERE Production.Product.Color IS NOT NULL

```

- Add a table to the report.
- Drag the fields **ProductID**, **Name** and **Color** to the table.
- Right-click on the upper part of the first column.
- Click **Text Box Properties** ⇒ **Interactive Sorting**.
- Fill it in like this:



- Apply the Interactive sorting to the other columns as well.
- Run the report.

Product ID	Name	Color
712	AWC Logo Cap	Multi
952	Chain	Silver
322	Chaining	Black
320	Chaining Bolts	Silver
321	Chaining Nut	Silver

- Save the report as **Report006.rdl**.

1.8.2 Dynamic Sorting Using Parameters

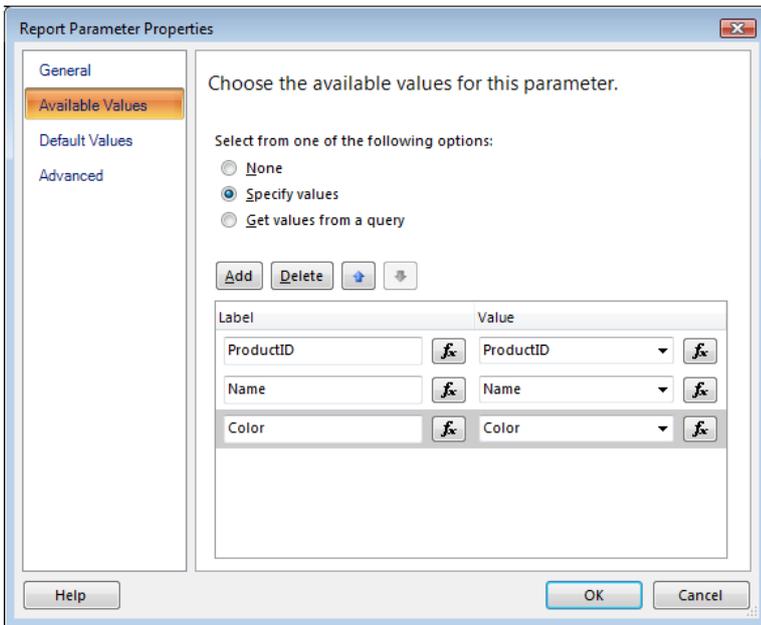
While that might make sense if everyone wants it that way, more than likely you might have people that want a report sorted differently by default. How to do it? There are probably a few ways, but here is a possibility.

First, we will add two parameters: **SortByDefault** and **SortOrder**. The **SortByDefault** will be a drop down of your columns you want to sort by for your dataset (or group, or table/tablix). The **SortOrder** is simply **Asc** (1 to N, A to Z) and **Desc** (N to 1, Z to A)

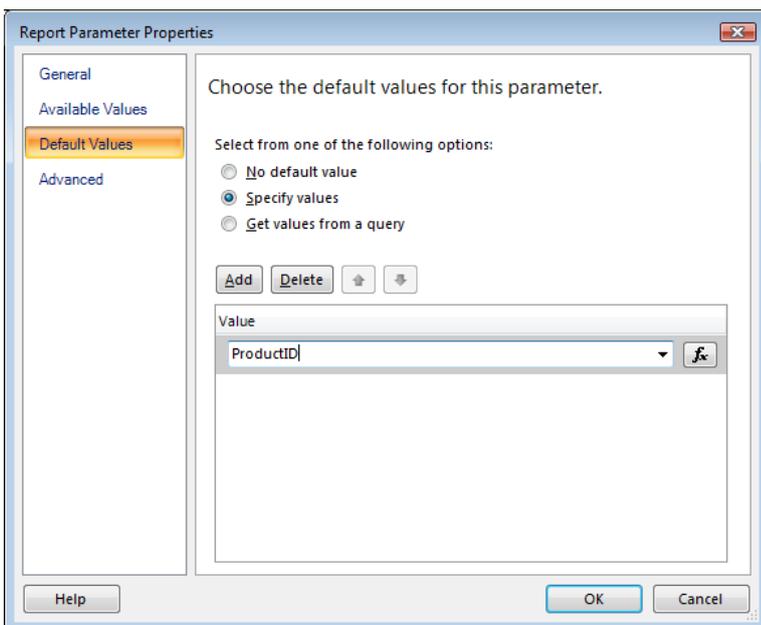
We'll continue with **Report006.rdl**.

- Right-click on **Parameter**.
- Click Add **Parameter**.
- Call the first one **SortByDefault**.

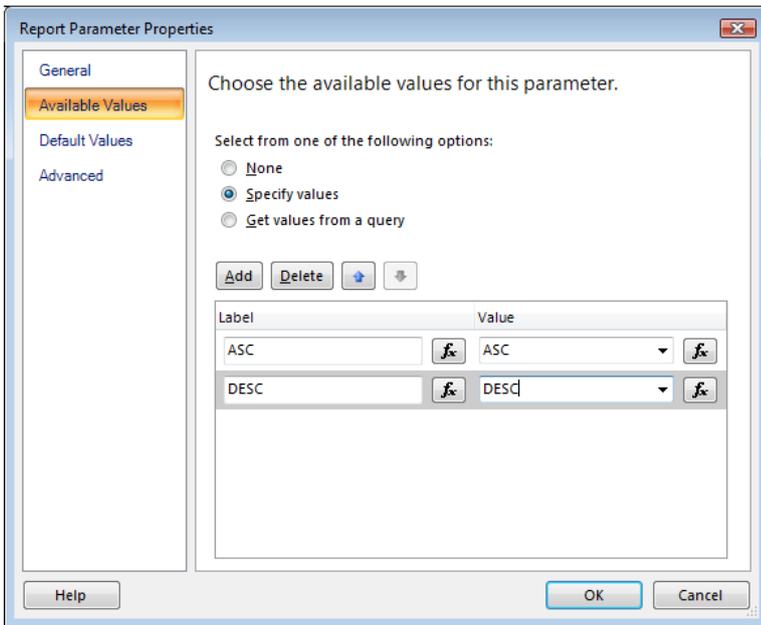
- Click **Available Values**.



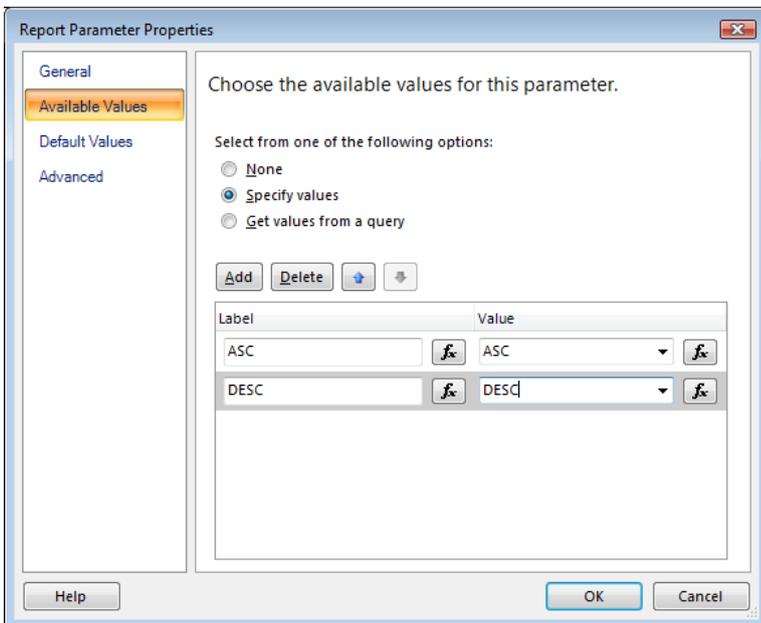
- Fill it in like here.
- Click **Default Value**.
- Click **Specify Values**.
- Click **Add**.
- Fill it in like this.



- Add another parameter **SortOrder** like this:



- And:

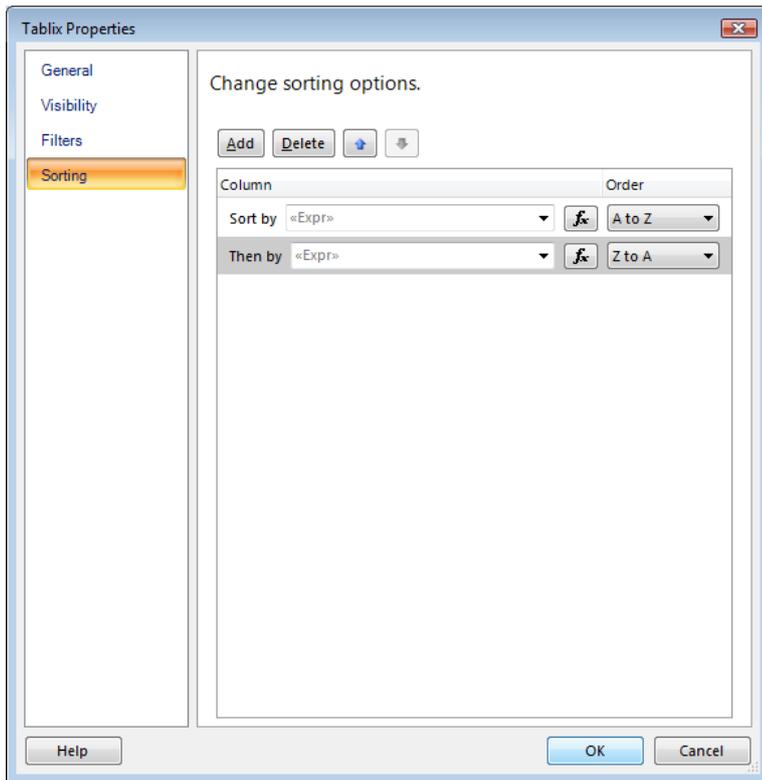


Now we will make it all work.

- Make sure you remove any **ORDER BY** in your dataset (you don't have to but this makes it easier).
- Right-click the table at the **left handler**.

Card Type	Card Number	Header
[CardType]	[CardNumber]	★ 👁️ 🗑️ 🔄 🖨️

- Click **tablix property**.
- Click **sorting**.



Now you can see, **Sort By** and **Then By** are expressions. Unfortunately, you can't set expressions for **ASC** or **DESC**. So we have to use a trick.

- The first is to handle the **ASC** option; A to Z;

```
=IIF (Parameters!SortOrder.Value="ASC", Fields (Parameters!SortByDefault.Value)
.Value, 0)
```

- The second is to handle the **DESC** option; Z to A!

```
=IIF (Parameters!SortOrder.Value="DESC", Fields (Parameters!SortByDefault.Value)
).Value, 0)
```

You can see, some magic. If the **Order By** is XYZ then use the field, otherwise 0. If you notice from the screenshot, **first one is A to Z (Asc) and the second one is Z to A (Desc)**. So we are basically telling **Report Builder** to sort by the parameter or not based on the order by option and it chooses the right **Order By (ASC/DESC)**.

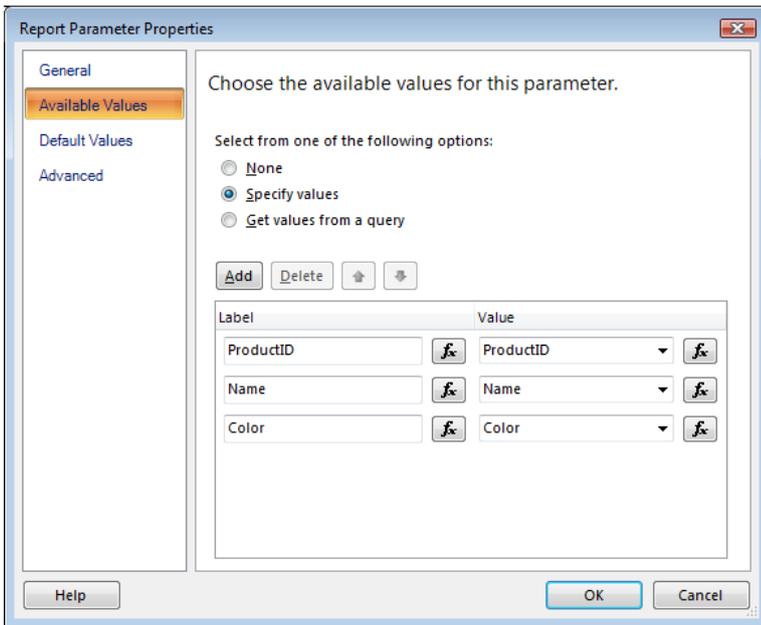
- Save the report (**Report006.rdl**).

1.8.3 Dynamic Sorting through the Dataset

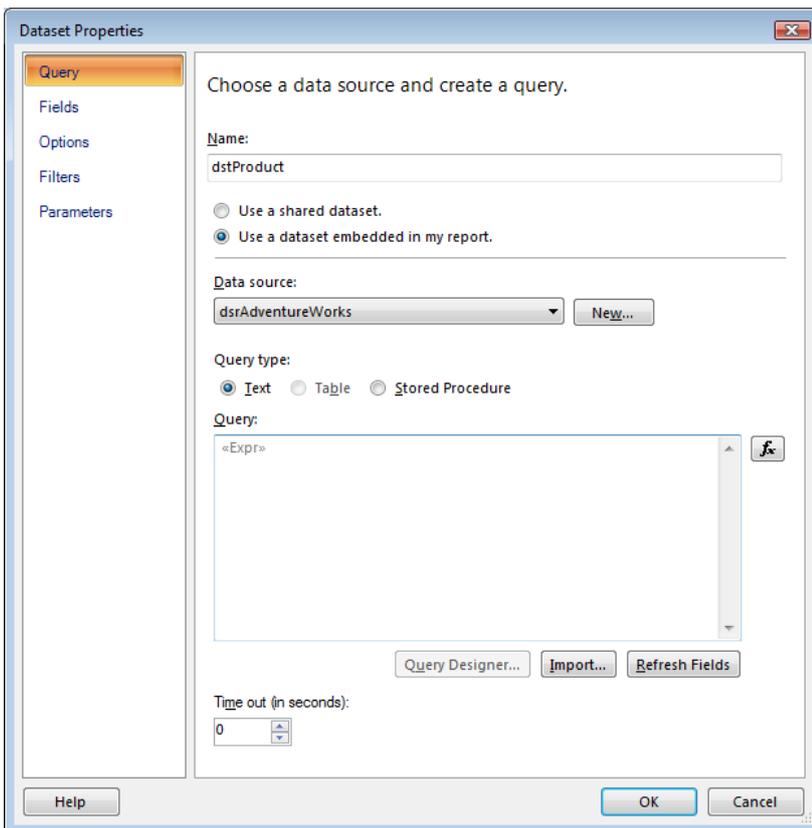
You can also dynamically sort a report through the dataset. In that case we have to create the dataset as an expression:

```
= " SELECT * FROM Production.Product ORDER BY " & Parameters!MySort.Value
```

- Create a new report.
- Save it as **Report007.rdl**.
- Add the data source **dsrcAdventureWorks**.
- Add a **Parameter MySort** with **Available Values**:

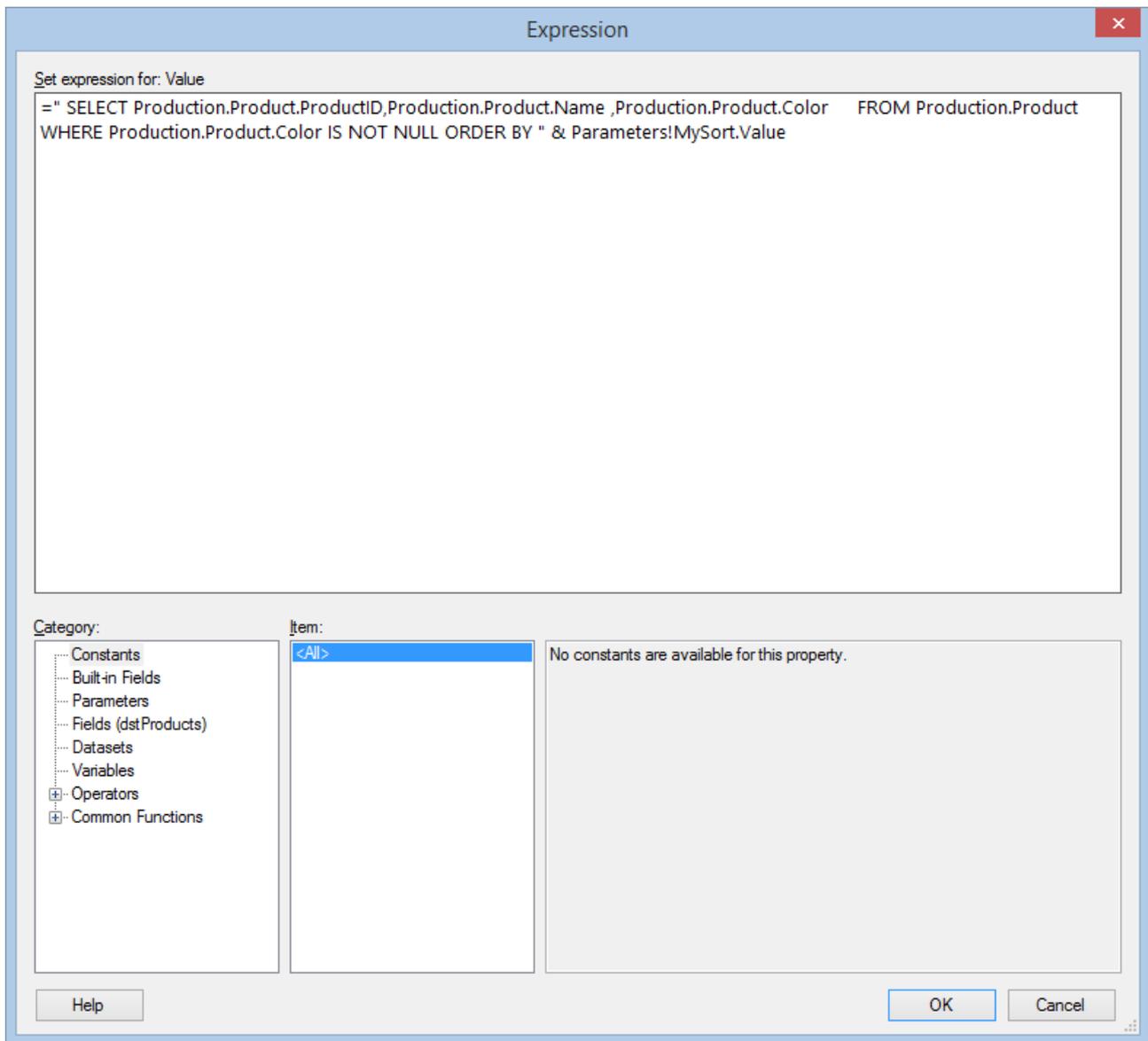


- Add **Default Values ProductID**.
- Add the dataset **dstProduct** by using the manual dataset construction.



- Click **fx**.

Before we fill in the expression in the picture, we have to fill in the regular **SELECT**. Otherwise, we will not get any fields.



- Fill in:

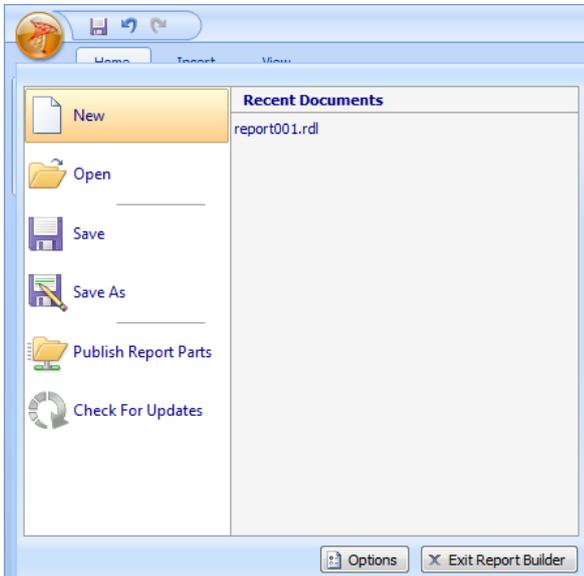
```
" SELECT Production.Product.ProductID,Production.Product.Name
,Production.Product.Color FROM Production.Product WHERE
Production.Product.Color IS NOT NULL ORDER BY " & Parameters!MySort.Value
```

- Add a table to the report.
- Drag the fields **ProductID**, **Name** and **Color** to the table.
- Run the report.
- Save the report (**Report007.rdl**).

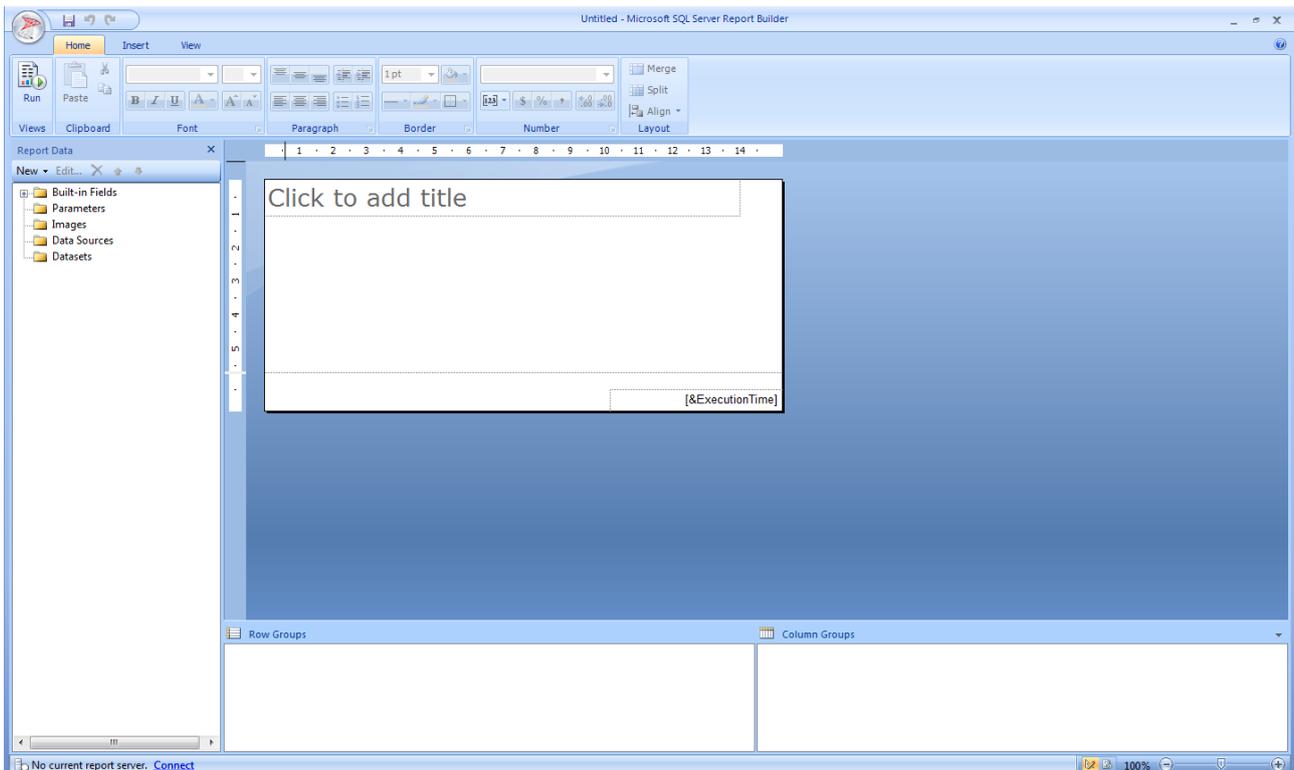
2 Designing the Report

2.1 Designing a Report Manually

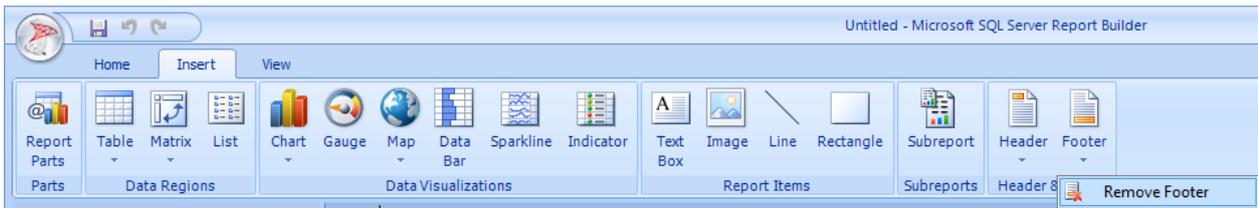
- Click New.



- Click **Blank Report**.



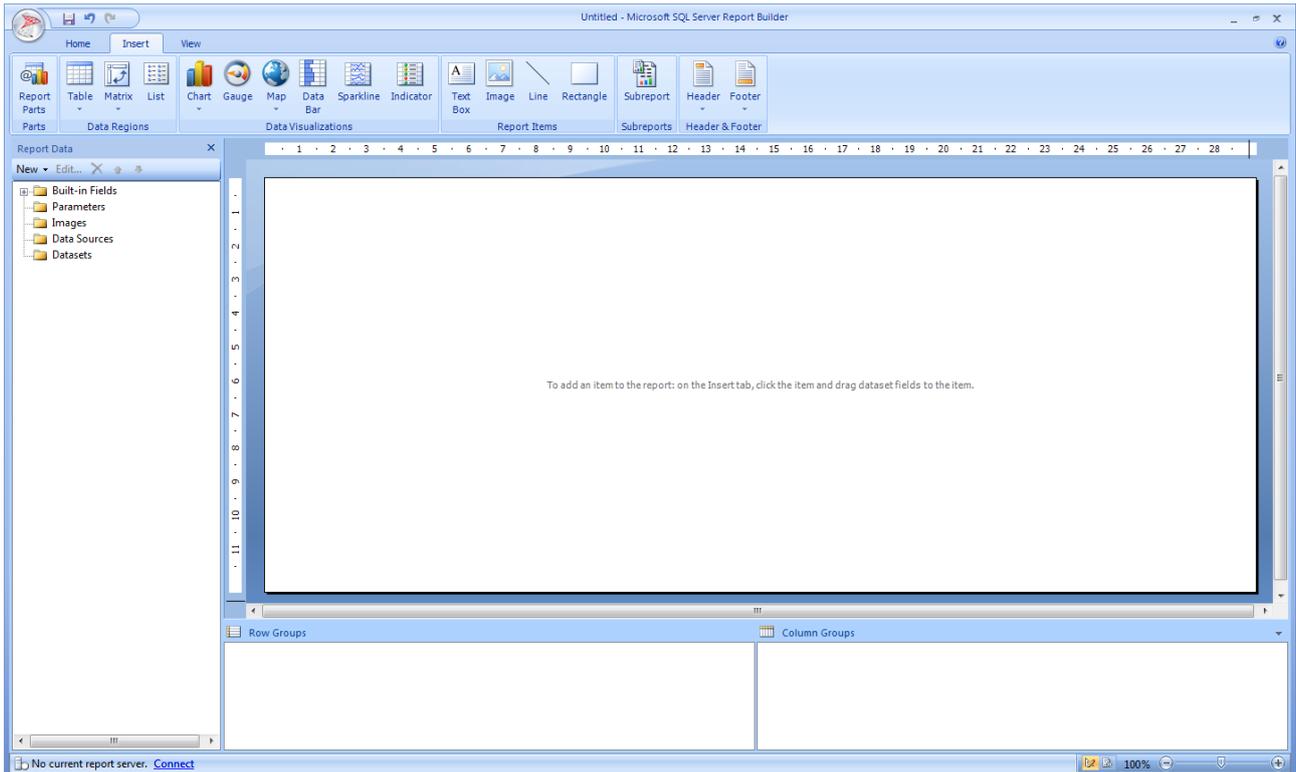
- Click on the tab **Insert**.
- Click **Remove Footer**.



- Also delete the **Click to add title** box.

Now we have an empty report.

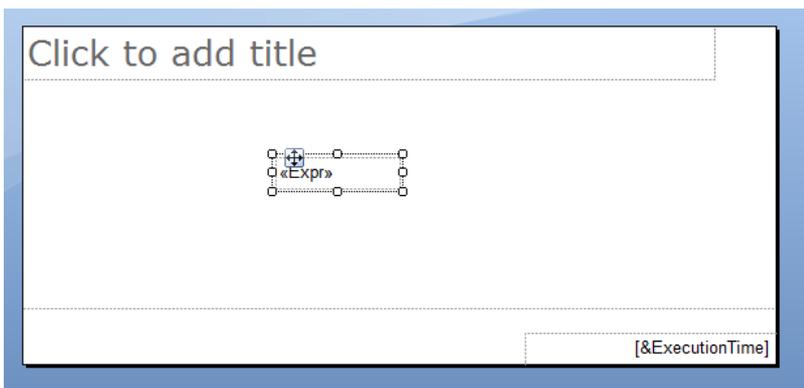
- To resize the canvas just drag a border to the desired shape.



- Close the report again without saving it.

2.2 Dragging fields directly to the report.

- Create a new, blank report.
- Create the data source **dsrcAdventureWorks**.
- Create the dataset **dstProduct**
- Drag one of the fields to the report canvas.



- Now on the **Home** tab click **Run**.

The content of the field will appear. Actually we are looking now at the content of a field of the first record of the table. Not very useful.

- Close the report without saving it.

2.3 Data Regions

2.3.1 Introduction

Tables, matrices, and lists are data regions that display report data in cells that are organized into rows and columns. The cells typically contain text data such as text, dates, and numbers but they can also contain gauges, charts, or report items such as images. Collectively, tables, matrices, and lists are frequently referred to as tablix data regions.

The table, matrix, and list templates are built on the tablix data region, which is a flexible grid that can display data in cells. In the table and matrix templates, cells are organized into rows and columns. Because templates are variations of the underlying generic tablix data region, you can display data in combination of template formats and change the table, matrix, or list on to include the features of another data region as you develop your report. For example, if you add a table and find it does not serve your needs, you can add column groups to make the table a matrix.

The table and matrix data regions can display complex data relationships by including nested tables, matrices, lists, charts and gauges. Tables and matrices have a tabular layout and their data comes from a single dataset, built on a single data source.

The key difference between tables and matrices is that tables can include only row groups, whereas matrices have row groups and column groups. You can compare matrices to pivot tables or cross tabs.

Lists are a little different. They support a free-layout that can include multiple peer tables or matrices, each using data from a different dataset. Lists can also be used for forms, such as invoices.

Note: You can publish tables, matrices, and lists separately from a report as report parts. Report parts are self-contained report items that are stored on the report server and can be included in other reports. Use Report Builder to browse and select parts from the Report Part Gallery to add to your reports. Use Report Designer or **Report Builder** to save report parts for use in the Report Part Gallery.

2.3.2 Table

2.3.2.1 Introduction

Use a table to display detail data, organize the data in row groups, or both. The table template contains three columns with a table header row and a details row for data. The following figure shows the initial table template, selected on the design surface:

	Header	
	Data	

You can group data by a single field, by multiple fields, or by writing your own expression. You can create nested groups or independent, adjacent groups and display aggregated values for grouped data, or add totals to groups. For example, if your table has a row group called [Category], you can add a subtotal for each group as well as a grand total for the report. To improve the appearance of the table and highlight data you want to emphasize, you can merge cells and apply formatting to data and table headings.

You can initially hide detail or grouped data, and include drilldown toggles to enable a user to interactively choose how much data to show.

2.3.2.2 Adding a Table to Display Detail Data

- Create a new, blank report.
- Save the report as **Report008.rdl**.
- Create the data source **dsrAdventureWorks**.
- Create the dataset **dstSales**, use the **SQL** query below (can be created menu driven):

```
SELECT OrderDate AS Date, SalesOrderHeader.SalesOrderID AS [Order], Name AS
Product, OrderQty AS Qty, LineTotal AS [Line Total]
FROM
Sales.SalesOrderHeader INNER JOIN Sales.SalesOrderDetail
ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID
INNER JOIN Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID
```

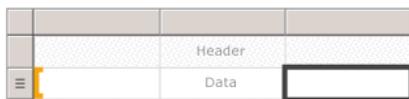
- Add a table to the design surface from the **Insert** tab on the ribbon.

You can add a table by using the **Table** or **Matrix Wizard**, which includes creating a data source connection and dataset and configuring the table, or a table based on the table template, which you configure manually.

To describe how to configure a table from beginning to end, this topic uses the table template. By default, a new table has a fixed number of columns with a header row for labels and a data row for detail data. The following figure shows a new table added to the design surface.



When you select the table, row and column handles appear on the outside of the table and brackets appear inside cells. Row handles display graphics that help you understand the purpose of each row. Brackets indicate group membership for a selected cell. The following figure shows a selected empty cell in a default table.



The row handle for the Data row shows the details symbol (☰).

- To display data on these rows, drag the fields **Date**, **Order**, **Product**, **Qty**, and **Line Total** from the Report Data pane to the table cells in either the header or the details row.

Both rows are filled in simultaneously.

- To add additional columns, drag the field to the table until you see an insertion point.
- After you add dataset fields to the table, you can change the default format for dates and currency to control the way they display in the report.

The following diagram shows a table data region with these fields: **Date**, **Order**, **Product**, **Qty**, and **Line Total**.

	Date	Order	Product	Qty	Line Total
☰	[Date]	[Order]	[Product]	[Qty]	[Line Total]

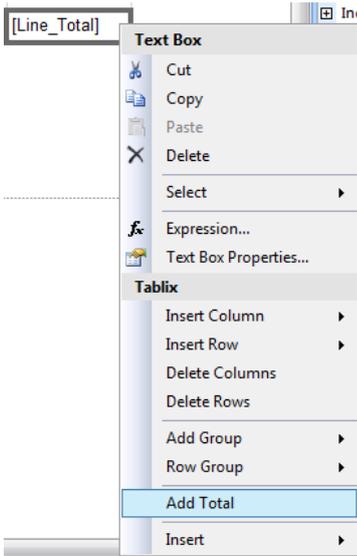
- Check your design by viewing the report in Preview.

The table expands down the page as needed. The label row and the details row each display once for every row in the dataset query result set. Each product sold in the order is listed on a separate row, along with the quantity and the line total for the item, as shown in the following figure:

Date	Order	Product	Qty	Line Total
July 01, 2005	SO43659	AWC Logo Cap	2	\$10.37
July 01, 2005	SO43659	Long-Sleeve Logo Jersey, M	3	\$86.52
July 01, 2005	SO43659	Long-Sleeve Logo Jersey, XL	1	\$28.84

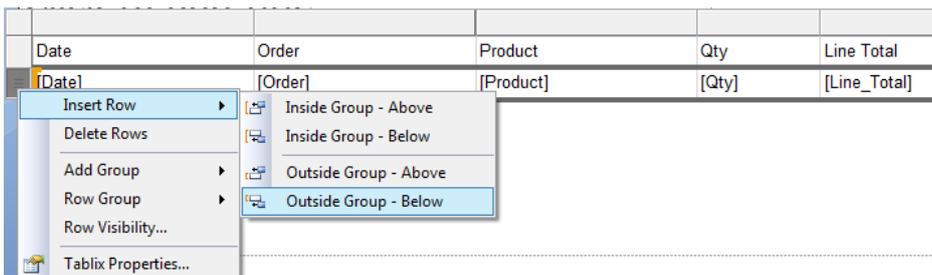
2.3.2.3 Adding Totals for Detail Data

- To add totals, select cells with numeric data.
- Right click a cell.
- Click **Add Total**.



You can also specify other labels and totals manually.

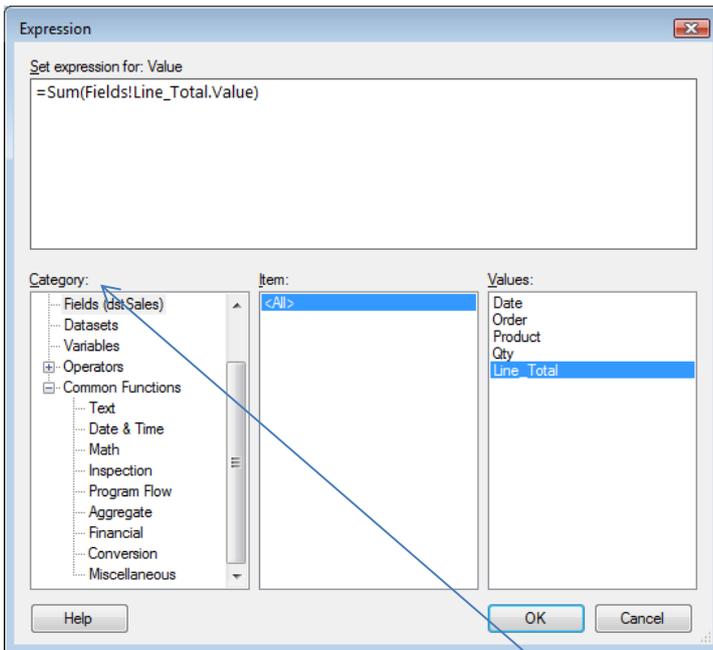
- Remove the Total row you just created by right clicking the row at the left.
- Click **Delete Rows**.
- Right click the row at the left again:



- Click **Insert Row**.
- Click **Outside Group - Below**.

A new row is added.

- Right click the last cell of the new row.
- Click **Expression**.



- Fill in the Formula (can be done through **Category**)

=Sum(Fields!Line_Total.Value, "dstSales")

In this case, **dstSales** is the range for which the **Line_Total** is summarized.

- Click **OK**.
- Run the report.
- Go to the last page.
- Check the total.

The following figure shows a typical totals row that includes both automatic and manually specified totals:

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
Total Orders:	[CountDistinct(Order)]	Total:	[Sum(Qty)]	[Sum(LineTotal)]

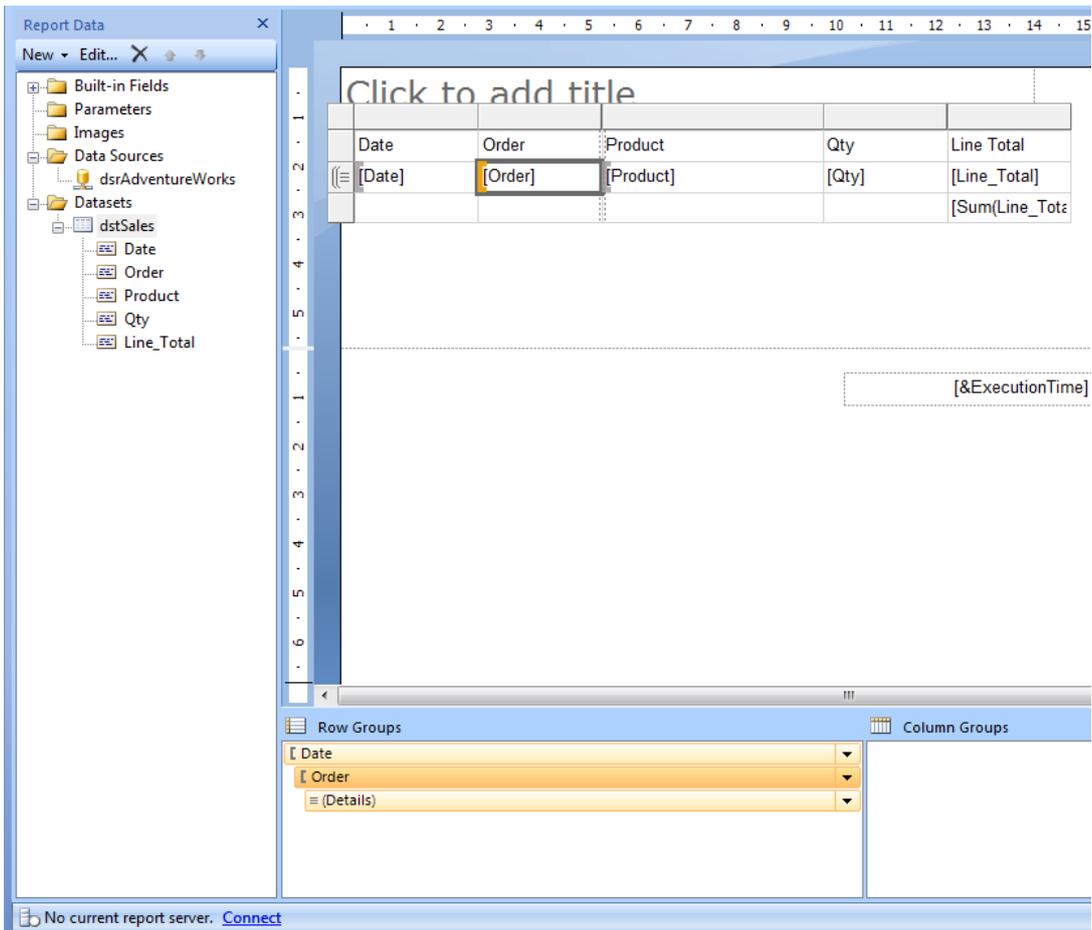
In **Preview**, the report displays the header row and the details row once for every row in the dataset query result set, and it displays the totals row. The follow figure shows the last few rows of the table including the total row.

June 01, 2004	SO71952	Women's Mountain Shorts, M	3	\$125.98
June 01, 2004	SO71952	Women's Mountain Shorts, S	3	\$125.98
Total Orders:	2416	Total Sales:	\$1,780,769.91	

2.3.2.4 Adding Row Groups to a Table

Just like you can drag a field from the Report Data pane to a cell to display detail data, you can drag a field to the Grouping pane to add a group.

- Delete the columns **Date** and **Order**.
- Drag the fields **Date** and **Order** to the **Row Groups** pane, above **Details**.



After you add a group, the table automatically adds cells in new columns in the row group area in which to display the group values.

The following figure shows a table with two nested row groups in Design view. The row groups were created by dragging the Order field and then the Date field to the Row Groups pane and inserting each group as a parent of the existing groups. The figure shows a parent group based on date and a child group based on order number, as well as the details group that was defined by default.

	Date	Order	Product	Qty	Line Total
[[[Date]	[Order]	[Product]	[Qty]	[Line Total]

In Preview, the report displays the order data grouped first by date, and then by order, as shown in the follow figure.

Date	Order	Product	Qty	Line Total
July 01, 2005	SO43659	AWC Logo Cap	2	\$10.37
		Long-Sleeve Logo Jersey, M	3	\$86.52
		Long-Sleeve Logo Jersey, XL	1	\$28.84
		Mountain Bike Socks, M	6	\$34.20
	SO43661	AWC Logo Cap	4	\$20.75
		Long-Sleeve Logo Jersey, L	4	\$115.36
		Long-Sleeve Logo Jersey, XL	2	\$57.68

2.3.2.5 Adding Totals to Row Groups

To show totals for a group, you can use the context-sensitive **Add Total** command. For a row group, the **Add Total** command adds a row outside the group so that it repeats only once in relation to the group. For nested groups, the total row for the child group is outside the child group but inside the parent group. In such a case, it is useful to set the background color of the total row for the child group to distinguish it from the detail rows. You can also use a different background color to distinguish the table header and footer rows.

- Right click the fields **Qty** and **Line total**.
- Click **Add Total**.

The following figure shows the table with a total row added for the group based on order numbers.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
	CountDistinct(Order)	Subtotal:	Sum(Qty)	Sum(LineTotal)
Total Orders	CountDistinct(Order)	Total:	Sum(Qty)	Sum(LineTotal)

When you view the report, the row displaying the order subtotals repeats once for every order number. The table footer displays totals for all dates. In the following figure, the last few rows show the last three detail rows, the subtotal for the last order number SO71952, and the totals for all dates in the table.

	S071952	Women's Mountain Shorts, L	15	\$548.55
		Women's Mountain Shorts, M	3	\$125.98
		Women's Mountain Shorts, S	3	\$125.98
	111	Subtotal:	2999	\$83,642.49
Total Orders:	2416	Total:	64569	\$1,780,769.91

- To add a subtotal for the Date group, right click the subtotals for the group Order.
- Click **Add total** again.

2.3.2.6 Removing or Hiding Detail Rows

After you preview a table in a report, you may decide to remove existing detail rows. Or you might decide to hide them by default and allow the user to toggle between viewing more or less detail, as in a drilldown report.

- To remove detail rows from a table, use the Grouping pane.
- Select the detail group.
- Use the shortcut menu to delete the group and the rows that display the detail data.

The following figure shows the design view for a table grouped by date and order number, but with no detail rows. No total rows have been added to this table.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Sum(Qty)]	[Sum(LineTotal)]

After you delete the details row, values are scoped to the row groups. The detail data no longer displays.

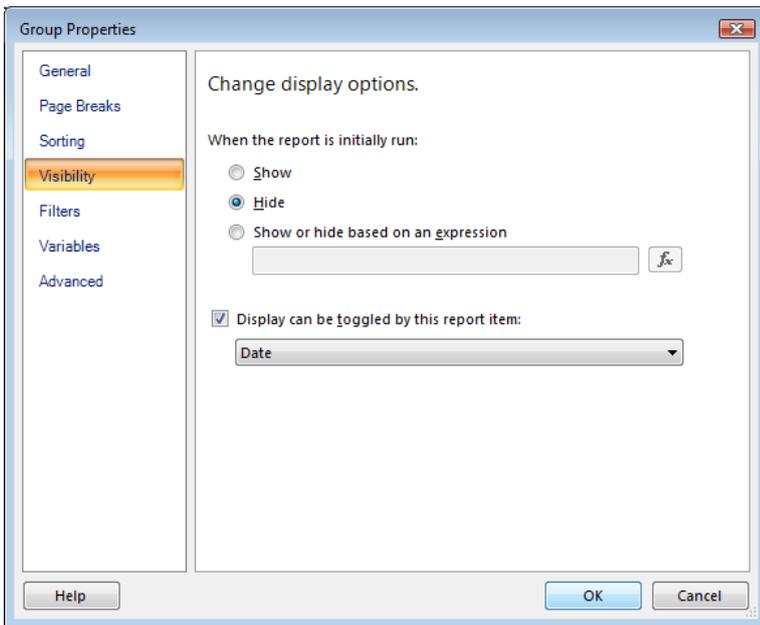
The following figure shows this report in Preview.

Date	Order	Product	Qty	Line Total
July 01, 2001	S043659	AWC Logo Cap	12	\$159.93
	S043659	AWC Logo Cap	10	\$193.79
	S043659	Long-Sleeve Logo Jersey M	2	\$57.68
	S043659	AWC Logo Cap	10	\$102.25

You can also hide the detail rows when the report is initially viewed.

To do so, you can create a drilldown report, in which only the parent group data is displayed. For each inner group (including the details group), add a **visibility toggle** to the grouping cell of the containing group.

- Right click **Details** in the **Row groups**.
- Click **Group properties**.
- Click **Visibility**.



- Click **OK**.
- Run the report.

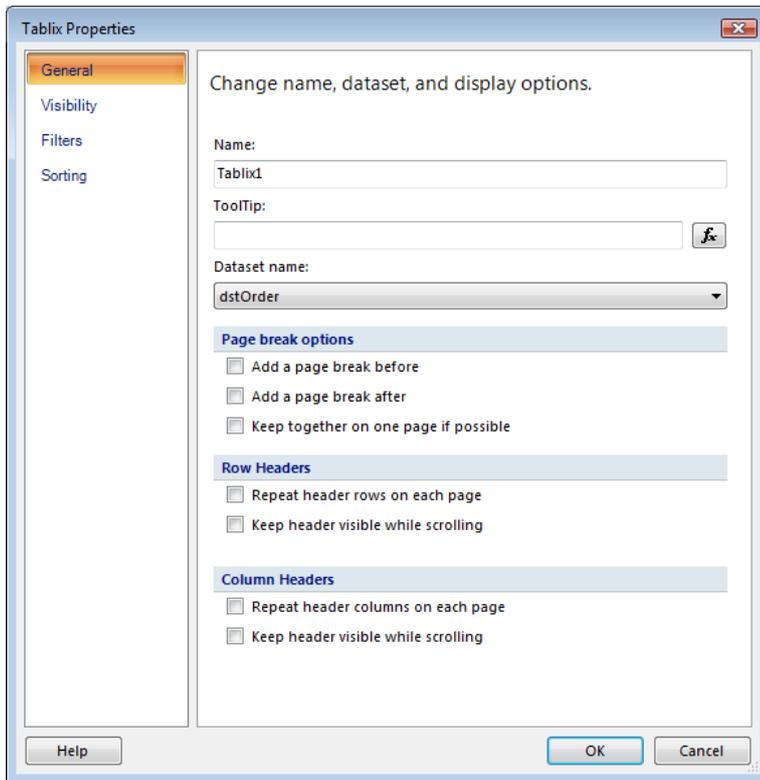
The following figure shows the row for September 01, 2001, expanded to display the first few orders.

Date	Order	Product	Qty	Line Total
July 01, 2001		Mountain Bike Socks, M	167	\$2,875.15
August 01, 2001		AWC Logo Cap	449	\$7,038.58
September 01, 2001	S044074	AWC Logo Cap	1	\$5.19
	S044075	Mountain Bike Socks, L	1	\$5.70
		Mountain Bike Socks, M	2	\$11.40

- Save the report (**Report008.rdl**).

2.3.2.7 Table properties

- By right clicking the upper left corner of a table, we'll get the properties:



Row headers

In a table, for example, you do not have column groups, so **Report Builder** does not recognize which textboxes are the column headers and checking **Repeat header columns on each page** doesn't work. A known bug for which we have a work around.

- Create a new report.
- Save the report as **Report009.rdl**.
- Data source **dswAdventureWorks**.
- Dataset **dstOrder**.

SELECT

```

Sales.SalesOrderHeader.SalesOrderID
, Sales.SalesOrderHeader.TotalDue
FROM
Sales.SalesOrderHeader

```

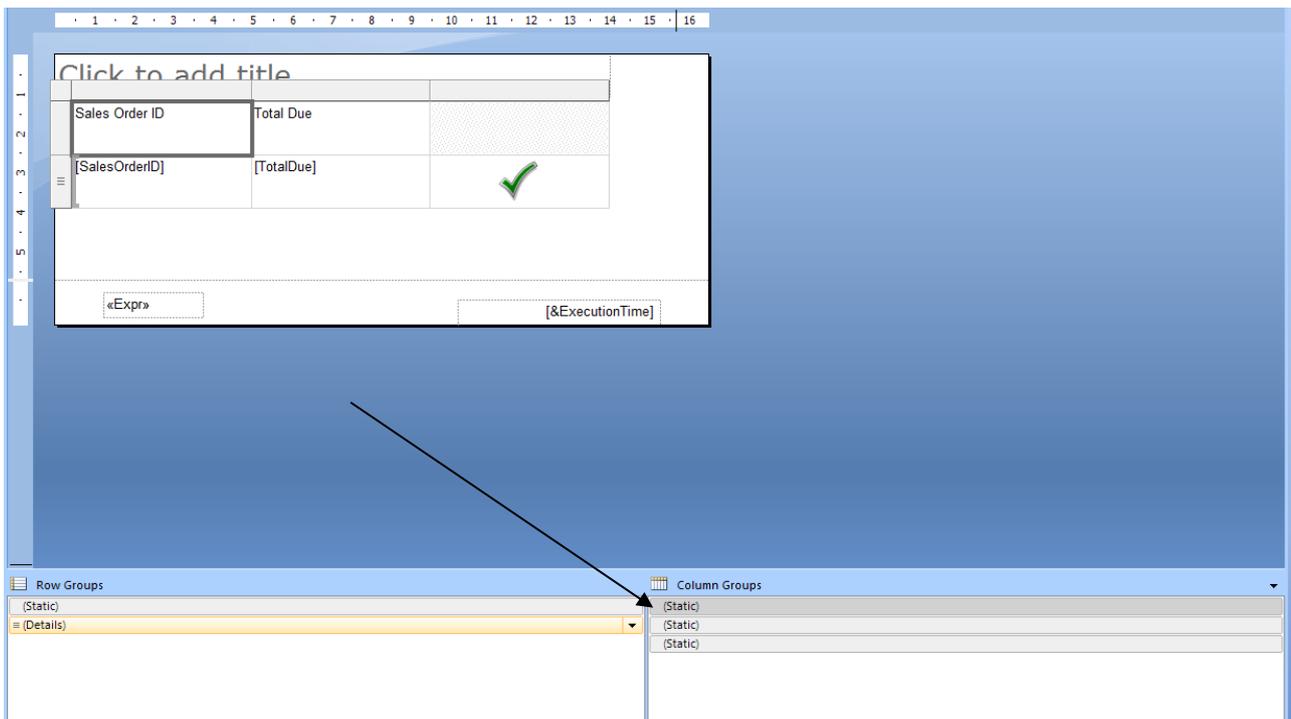
- Add a table to the blank report
- Add both fields to the table.

Instead, we need to open **Advanced Mode** in the **Groupings** pane:



- Click the arrow to the right of the **Column Groups**
- Select **Advanced Mode**.

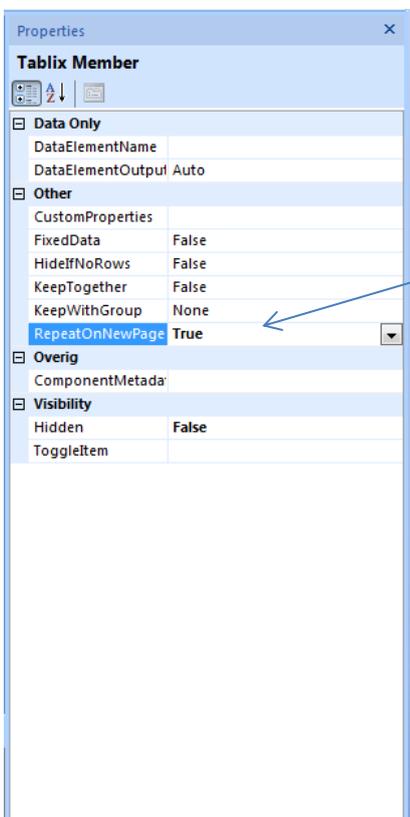
You'll see **Static Groups** appear in the **Row Groups** area.



- Clicking on a **Static group** highlights the corresponding textbox in the tablix.
- For the column headers that you want to repeat, select the **Static group** that highlights the leftmost column header.

This is generally the first **Static group** listed in **Row Groups**.

- In the Properties pane, set the **RepeatOnNewPage** property to **True**.
- Make sure that the **KeepWithGroup** property is set to **After**.



The **KeepWithGroup** property specifies which group to which the static member needs to stick. If set to **After** then the static member sticks with the group after, or below, it acting as a group header. If set to **Before**, then the static member sticks with the group before, or above it, acting as a group footer. If set to **None**, **Report Builder** decides where to put the static member.

Now when you view the report, the column headers repeat on each page of the tablix.

- Save the report (**Report009.rdl**).

2.3.3 Matrix

2.3.3.1 Introduction

Use a matrix to display aggregated data summaries, grouped in rows and columns, similar to a PivotTable or crosstab. The number of rows and columns for groups is determined by the number of unique values for each row and column groups. The following figure shows the initial matrix template, selected on the design surface:



You can group data by multiple fields or expressions in row and column groups. At run time, when the report data and data regions are combined, a matrix grows horizontally and vertically on the page as columns for column groups and rows for row groups are added.

The matrix cells display aggregate values that are scoped to the intersection of the row and column groups to which the cell belongs. For example, if your matrix has a row group (**Category**) and two column groups (**Territory** and **Year**) that display the sum of sales, the report displays two cells with sums of sales for each value in the Category group. The scope of the cells are the two intersections are: **Category** and **Territory** and **Category** and **Year**.

The matrix can include nested and adjacent groups. Nested groups have a parent-child relationship and adjacent groups a peer relationship. You can add subtotals for any and all levels of nested row and column groups within the matrix.

To make the matrix data more readable and highlight the data you want to emphasize, you can merge cells or split horizontally and vertically and apply formatting to data and group headings.

You can also include drilldown toggles that initially hide detail data; the user can then click the toggles to display more or less detail as needed.

2.3.3.2 Adding a Matrix to Your Report

- Create a new, blank report.
- Save it as **Report010.rdl**.
- Create the data source **dsrcAdventureWorks**.
- Create the dataset **dstSalesTerritory**, use the **SQL** query below:

```

SELECT Production.ProductCategory.name AS Category,
Production.ProductSubcategory.name AS Subcat, LineTotal as [Line Total],
CountryRegionCode as [Country Region], [Group] as Geography, ,
YEAR(OrderDate) AS [Year]
FROM
Sales.SalesOrderHeader
INNER JOIN Sales.SalesOrderDetail
ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID
INNER JOIN Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID
INNER JOIN Production.ProductSubcategory
on Production.ProductSubcategory.ProductSubcategoryID=
Production.Product.ProductSubcategoryID
INNER JOIN Production.ProductCategory
ON
Production.ProductCategory.ProductCategoryID=Production.ProductSubcategory.P
roductcategoryID
INNER JOIN Sales.SalesTerritory on Sales.SalesTerritory.TerritoryID =
Sales.SalesOrderHeader.TerritoryID

```

- Add a matrix to the design surface from the **Insert** tab on the ribbon.

You have the option to add a matrix by using the **Table** or **Matrix** Wizard, which includes creating a data source connection and dataset, and configuring the matrix or adding a matrix based on the matrix template.

To describe how to configure a table from beginning to end, this topic uses the matrix template. The matrix initially has a row group, a column group, a corner cell, and a data cell, as shown in the following figure.

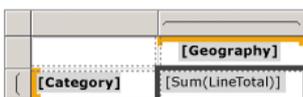


When you select a matrix on the design surface, row and column handles appear, as shown in the following figure.



Add groups by dragging dataset fields to the **Row Groups** and **Column Groups** areas of the **Grouping** pane. The first field that you drag to the row groups or column groups pane replaces the initial empty default group. You can then apply formatting for each cell, depending on the data.

- Drag the field **Geography** to the **Columns Groups**.
- Drag the field **Category** to the **Rows Groups**.
- Drag the fields **Line total** to the **Data** cell in the matrix.



In **Preview**, the matrix expands to show the row group and column group values. The cells display summary values, as shown in the following figure.

	North America	Europe	Pacific
Accessories	\$384,057	\$148,057	\$18,599
Bikes	\$51,990,891	\$11,503,109	\$1,186,388
Clothing	\$1,251,999	\$443,035	\$33,683
Components	\$8,601,244	\$2,746,093	\$183,142

The matrix you start with is a template based on the tablix data region. You can continue to develop your matrix design by adding nested or adjacent row groups or column groups, or even adding detail rows.

2.3.3.3 Adding a Parent Group or Child Group to a Matrix

- To add a group based on a single dataset field, drag the field from the Report Data pane to the appropriate Row Groups or Column Groups area of the Grouping pane.
- Drop the field in the group hierarchy to set its relationship to existing groups.
- Drop it above an existing group to create a parent group, or drop it below an existing group to create a child group.

Several things happen when you drop a field in the **Grouping** pane:

- A new group with a unique name based on the field name is automatically created. The group expression is set to the simple field name reference, for example

■ **[Category]**

- A new row or column appears in the corresponding row group or column group area.
- In the new column, a row group cell appears for the default data rows from the report dataset. Cells in the tablix body for this row are now members of the row group. If there are any column groups defined, cells that are in the columns are members of those column groups. Group indicators provide visual cues for the group membership of each cell.

To customize the group after it is created, use the **Tablix Group** dialog box. You can change the group name, and edit or add additional expressions to the group definition.

- Right click **Category**.
- Click **Add Total**.
- Right click **Subcat**.
- Click **Add Total**.
- Right click **Geography**.
- Click **Add Total**.

		[Geography]	Total
		[CountryRegion]	
[Category]	[Subcat]	[Sum(LineTotal)]	[Sum(LineTotal)]
		[Sum(LineTotal)]	[Sum(LineTotal)]
Total		[Sum(LineTotal)]	[Sum(LineTotal)]

When the report runs, dynamic column headers expand right (or left, if the **Direction property** of the matrix is set to **RTL** (right to left)) for as many columns as there are unique group values. Dynamic rows expand down the page. The data that appears in the tablix body cells are aggregates based on the intersections of row and column groups, as shown in the following figure.

In preview, the report displays as in the following figure.

Sales by Area and Year		North America		2003	2004	Total
		CA	US			
Clothing	Caps	\$2,801.06	\$9,798.76	\$9,008.44	\$3,591.38	\$12,599.82
	Tights	\$8,233.90	\$51,391.51	\$59,625.41		\$59,625.41
	Total	\$11,034.96	\$61,190.27	\$68,633.85	\$3,591.38	\$72,225.23
Components	Chains	\$793.83	\$4,972.00	\$3,543.48	\$2,222.35	\$5,765.83
	Cranksets	\$18,515.99	\$105,118.64	\$72,889.77	\$50,744.86	\$123,634.62
	Touring Frames	\$196,141.85	\$767,130.33	\$603,497.08	\$359,775.10	\$963,272.17
	Total	\$215,451.67	\$877,220.97	\$679,930.33	\$412,742.30	\$1,092,672.63
Grand Total	\$226,486.63	\$938,411.23	\$748,564.18	\$416,333.68	\$1,164,897.86	

To write expressions that specify a scope other than the default scope, you must specify the name of a dataset, data region, or group in the aggregate function all. To calculate the percentage each subcategory contributes to the Clothing category group values, add a column inside the Category group next to the Total column, format the text box to show percentage, and add an expression that uses the default scope in the numerator, and the Category group scope in the denominator, as shown in the following example.

=SUM(Fields!Linetotal.Value)/SUM(Fields! Linetotal.Value,"Category")

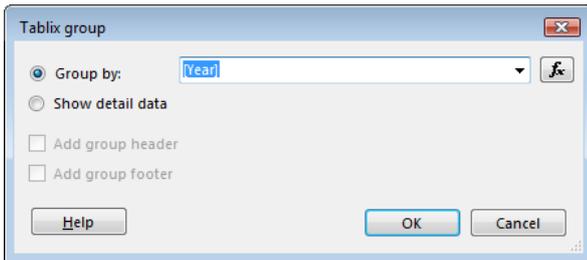
- Save the report (**Report010.rdl**).

2.3.3.4 Adding an Adjacent Group to a Matrix

To add an adjacent group based on a single dataset field, use the shortcut menu in the **Grouping** pane.

We'll continue with **Report010.rdl**.

- Right click **Geography**.
- Click **Add Group ⇒ Adjacent Right**.



- Pick **Year**.
- Drag the field **Line total** to the cell below **Year**.

The following figure shows a group based on geography and an adjacent group based on year.

		[Geography]	[Year]	Total
		[CountryRegion]		
[Category]	[Subcat]	[Sum(LineTotal)]	[Sum(LineTotal)]	[Sum(LineTotal)]

In this example, the query has filtered data values to only include those values for Europe and for the years 2003 and 2004. However, you can set filters on each group independently. In preview, the report displays as in the following figure.

		Europe			2003	2004	Total
		DE	FR	GB			
Accessories	Bike Racks	\$14,223	\$20,563	\$22,843	\$32,637	\$24,991	\$57,629
	Bottles and Cages	\$758	\$636	\$754	\$1,300	\$848	\$2,148
	Cleaners	\$860	\$1,022	\$1,323	\$1,936	\$1,269	\$3,206
	Helmets	\$10,661	\$12,587	\$20,495	\$28,389	\$15,804	\$44,193
	Hydration Packs	\$6,787	\$5,750	\$7,209	\$11,257	\$8,488	\$19,745

- To add a total column for an adjacent column group, click in the column group definition cell.
- Use the **Add Total** command.

A new static column is added next to the column group, with a default aggregate sum for every numeric field in the existing rows. To change the expression, manually edit the default aggregate, for example:

Avg([Sales]).

2.3.4 List

2.3.4.1 Introduction

Use a list to create a free-form layout. You are not limited to a grid layout, but can place fields freely inside the list. You can use a list to design a form for displaying many dataset fields or as a container to display multiple data regions side by side for grouped data. For example, you can define a group for a list; add a table, chart, and image; and display values in table and graphic form for each group value, as you might for an employee or patient record.

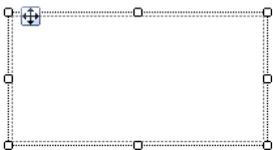
2.3.4.2 Adding a List to Your Report

- Create a new report.
- Save it as **Report011.rdl**.
- Add data source **dsrAdventureWorks**.
- Add dataset **dstProduct**:

```
SELECT OrderDate AS Date,SalesOrderHeader.SalesOrderID AS [Order],Name AS
Product, OrderQty AS Qty, LineTotal AS [Line Total], ThumbnailPhoto AS Photo
FROM
Sales.SalesOrderHeader
INNER JOIN Sales.SalesOrderDetail
ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID
INNER JOIN Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID INNER JOIN Production.ProductProductPhoto ON
Production.Product.ProductID = Production.ProductProductPhoto.ProductID
INNER JOIN Production.ProductPhoto ON
Production.ProductProductPhoto.ProductPhotoID =
Production.ProductPhoto.ProductPhotoID
WHERE Name like 'Mountain-100 Black%'
```

- Add a list to the design surface from the **Insert** tab on the ribbon.

By default, the list initially has a single cell in a row associated with the detail group.



- When you select a list on the design surface, row and column handles appear, as shown in the following figure.



The list you start with is a template based on the tablix data region. After you add a list, you can continue to enhance the design by changing the content or appearance of the list by specifying filter, sort, or group expressions, or changing the way the list displays across report pages.

Although the list starts with a single column and row, you can further continue to develop your list design by adding nested or adjacent row groups or column groups, or adding additional detail rows.

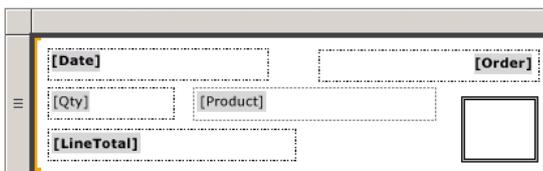
2.3.4.3 Displaying Data in a Free-form Layout

- To organize report data in a free-form layout instead of a grid, you can add a list to the design surface.
- Drag the fields **Date**, **Order**, **Qty**, **Product**, **LineTotal** from the Report Data pane to the cell.

By default, the cell contains a rectangle that acts as a container.

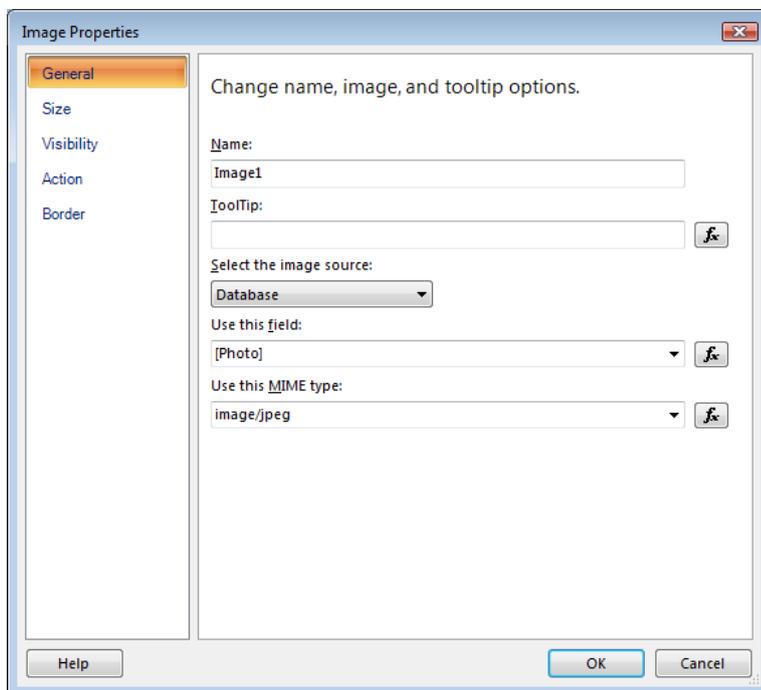
- Move each field in the container until you have the design you want.
- Use the snap lines that appear when you drag text boxes in the rectangle container to help you align edges vertically and horizontally.
- Remove unwanted white space by adjusting the size of the cell.

The following figure shows a list that displays information about an order, including these fields: **Date**, **Order**, **Qty**, **Product**, **LineTotal**, and an image.



To add a data-bound image:

- On the **Insert** menu, click **Image**.
- Draw a box in the container.
- Right-click the image box.
- Click **Image Properties**.
- On the **General** page of the **Image Properties** dialog box, type a name in the **Name** text box or accept the default.



- In **Select the image source**, select **Database**.
- In **Use this Field**, select the field that contains images in your report.
- In **Use this MIME type**, select the **MIME** type, or file format, of the image—for example, bmp.
- Click **OK**.

An image placeholder appears on the report design surface.

- Run the report.

In **Preview**, the list repeats to display the field data in the free-form format, as shown in the illustration below.

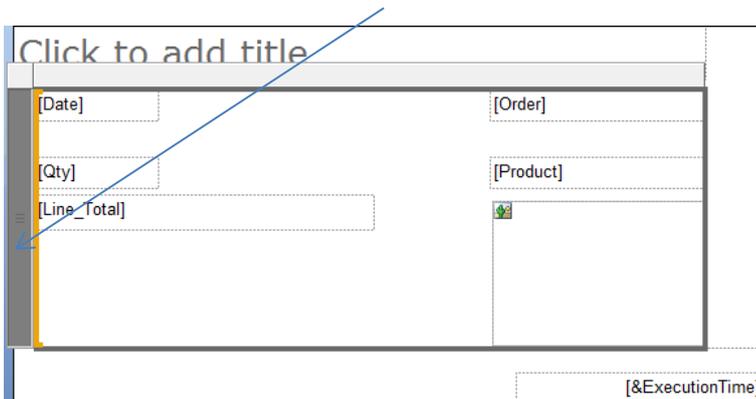


2.3.4.4 Displaying Data with One Level of Grouping

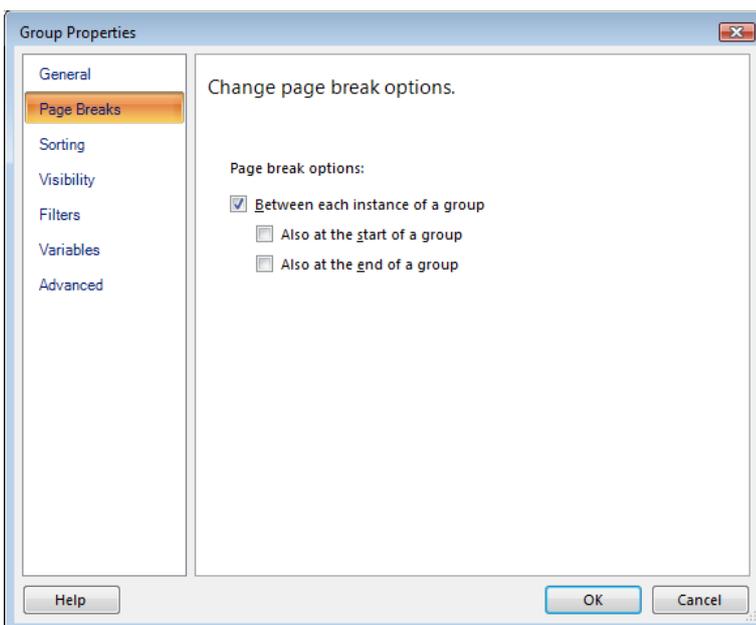
Because a list automatically provides a container, you can use a list to display grouped data with multiple views.

We'll continue with **Report011.rdl**.

- Right-click the list at the **row handle**.



- Click **Row Group** ⇒ **Group Properties**.



- To change the default list to specify a group, edit the **Details** group.
- Click **Page Breaks**.

- Check **Between each instance of a group**.
- Run the report.
- Save the report (**Report011.rdl**).

2.4 Header & Footer

To add a page header or footer

- Create a new, blank report.
- Save the report as **Report012.rdl**.
- On the design surface, right-click the report.
- Point to **Insert**.
- Click **Header** or **Footer**.

Note: The Header and Footer options appear only when a header or footer is not already part of the report.

To configure a page header or footer

- On the design surface, right-click the page header or footer.
- Point to **Insert**.
- Then click one of the following items to add it to the header or footer area:
 - Textbox
 - Line
 - Rectangle
 - Image
- Right-click the page header.
- Click **Header Properties** to add borders, background images, or colors, or to adjust the width of the header.
- Then click **OK**.
- Right-click the page footer.
- Click **Footer Properties** to add borders, background images, or colors, or to adjust the width of the footer.
- Then click **OK**.

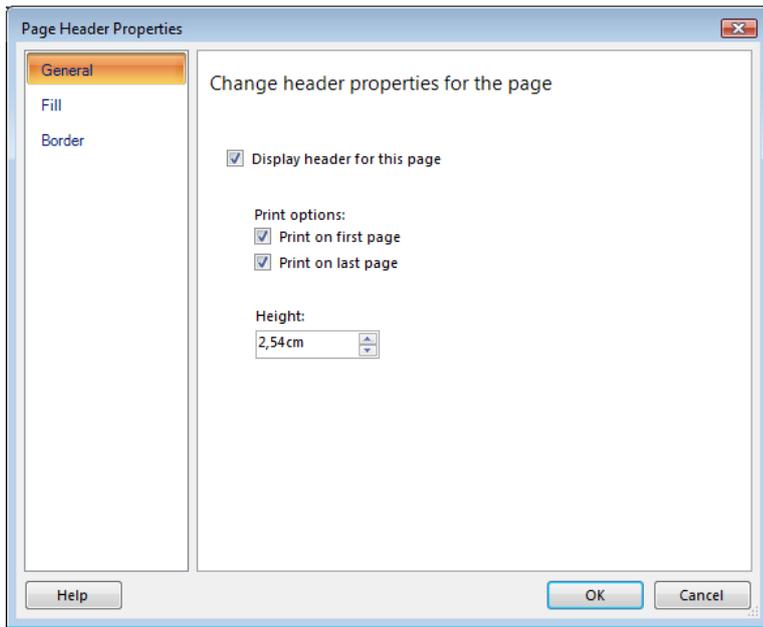
- On the design surface, right-click the page header or footer.
- Click **Remove Footer**.

Note: When you remove a page header or footer, you delete it from the report. Any items that you previously added to the page header or footer will not reappear if you subsequently add the header or footer again.

Report header

In the top of the body (read: top of the report) put everything you want to appear in the first page header.

In the page header section put everything that you want to appear on subsequent pages. Then set the page header to **NOT** print on the first page.



3 Formatting the Report

3.1 Cell formatting

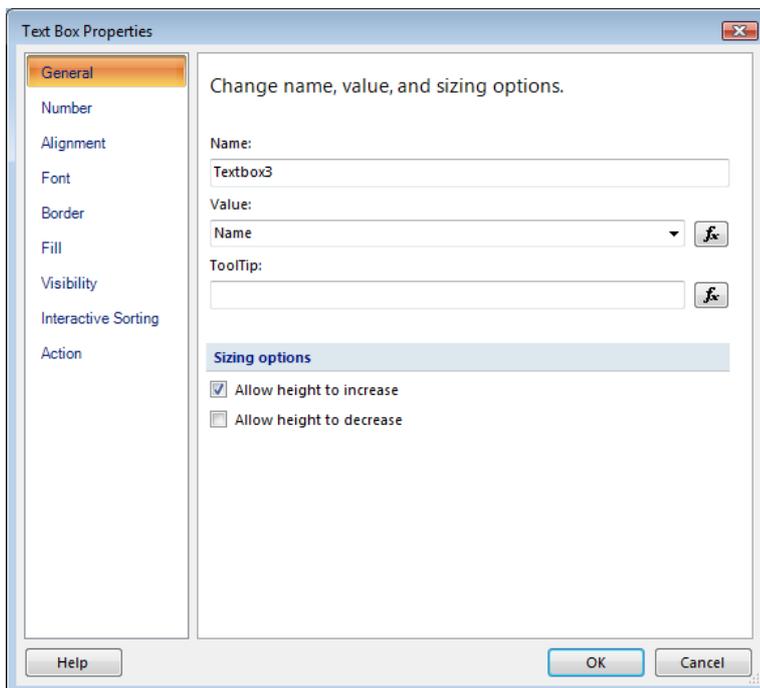
3.1.1 To Allow a Text Box to Grow or Shrink

- Create a new report.
- Save the report as **Report013.rdl**.
- Add data source **dsrAdventureWorks**.
- Add dataset **dstProduct**:

SELECT

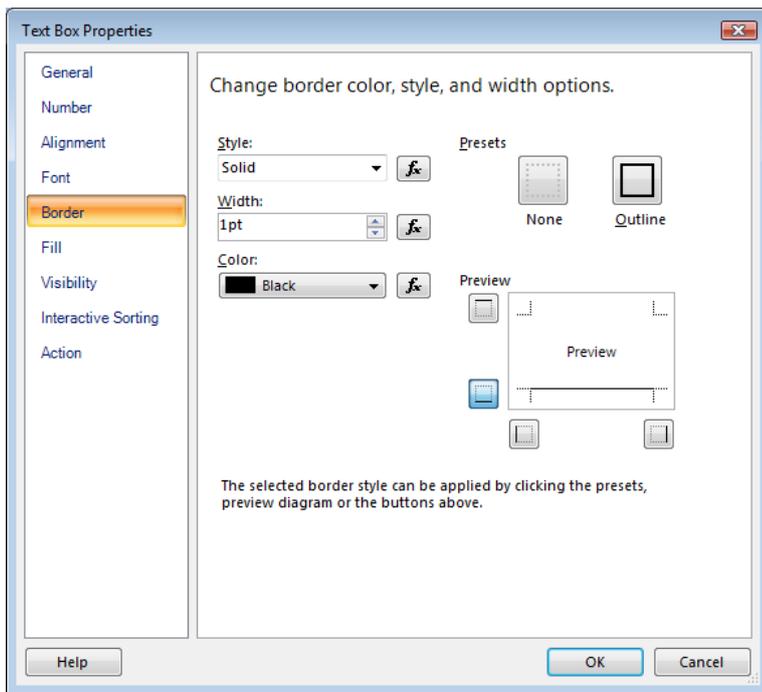
```
Production.Product.ProductID
,Production.Product.Name
,Production.Product.Color
,Production.Product.ListPrice
FROM
Production.Product
```

- Add a table to the report.
- Drag the field **ProductId**, **Name**, **Color** and **ListPrice** to the table.
- Right-click the text box **Name**.
- Click **Text Box Properties**.
- Click the **General** tab.
 - To allow the text box to expand vertically based on its contents, select **Allow height to increase**.
 - To allow the text box to shrink based on its contents, select **Allow height to decrease**.



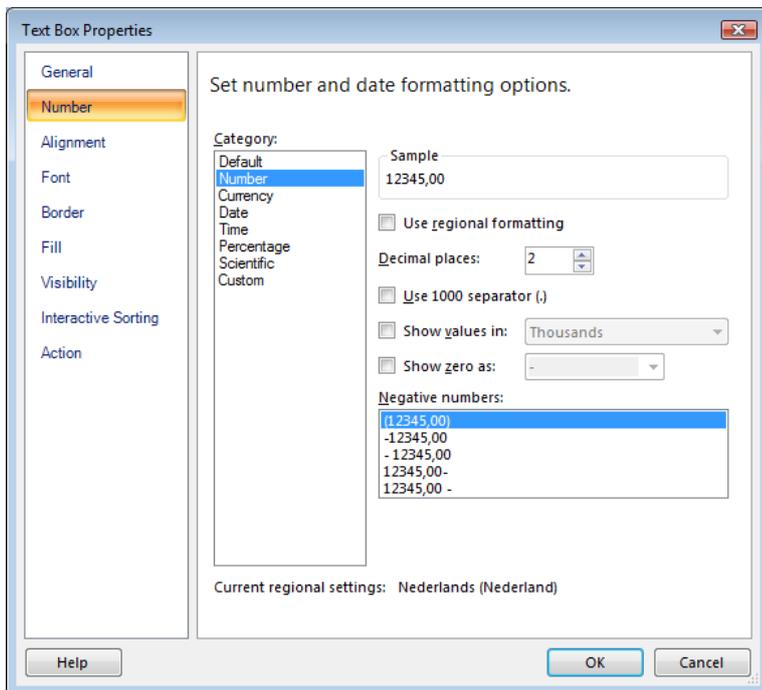
3.1.2 Underlining

- Right click on a cell in a table.
- Click **Text box Properties**.
- Click **Border**.



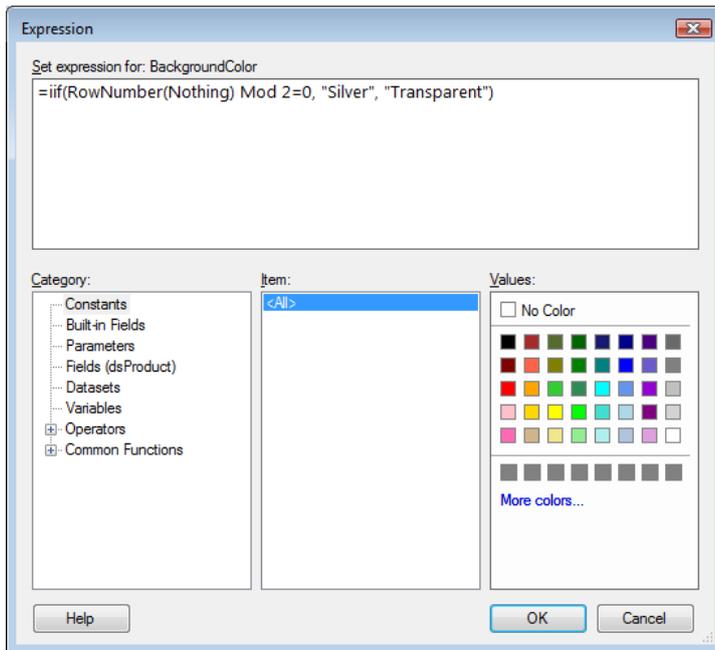
3.1.3 Numeric/currency

- Right-click **ListPrice** value.
- Click **Text Box Properties**.
- Click **Number**.



3.1.4 Alternate row coloring

- Select the lower row.
- In the **Properties pane**, choose **Fill** ⇒ **Background colors**.
- Click **Expression** after **Background colors**.



- Type the Formula

```
=IIf (RowNumber (Nothing) Mod 2=0, "Silver", "Transparent")
```

In a group the formula should look a bit different:

```
=IIf (RunningValue (Fields!Category.Value, CountDistinct, "NameGroup") Mod 2 = 0, "PaleGreen", "White")
```

"NameGroup" refers to the name of the specific group. When you fill in **Nothing** (without quotes) instead of "NameGroup" it refers to the top group.

- Save the report (**Report013.rdl**).

3.1.5 Displaying Checkboxes Instead of True/False

3.1.5.1 Introduction

It is nice to show images instead of for instance the content of a **Boolean** field (true or false). We will show two methods of accomplishing this.

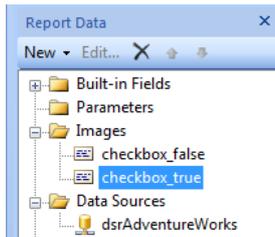
3.1.5.2 Method 1: Images

- Create a new report.
- Data source **dsrcAdventureWorks**.
- Dataset **dstCreditcard**:

```
SELECT
Sales.CreditCard.CreditCardID
,Sales.CreditCard.CardType
,Sales.CreditCard.CardNumber
,Sales.CreditCard.ExpMonth
,Sales.CreditCard.ExpYear
FROM
Sales.CreditCard
```

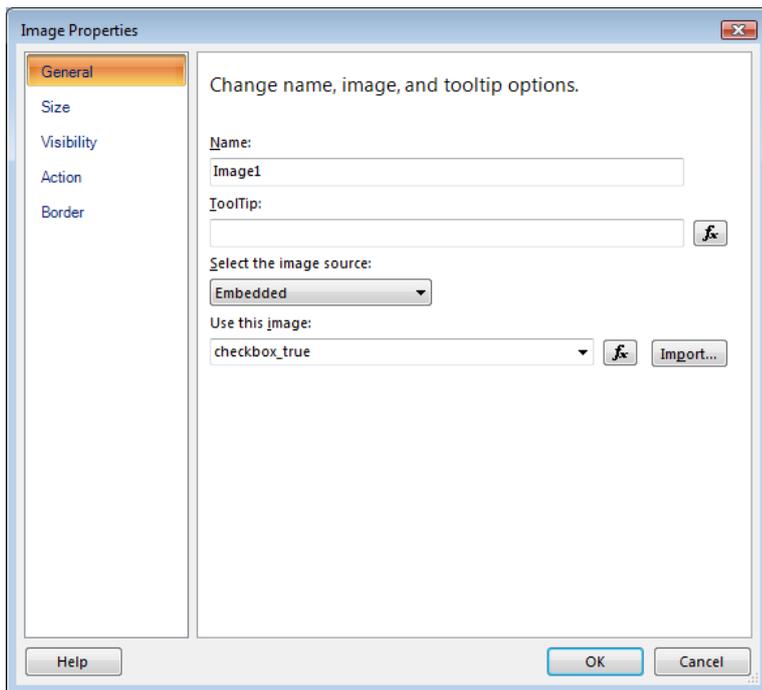
- Add two images by right clicking **Images**.
- Choose **Add image**.

- Pick the images **checkbox_false.jpg** and **checkbox_true.jpg**.



- Add a table to the new report.
- Add **Card Type** to the first column; **Card Number** to the second.
- Add the image **Checkbox_ true** to the third column.

We'll get this pop-up window:



- Clicking the **fx** button brings up the **Expression editor**, in which we create the following expression:

```
=IIF(Fields!ExpYear.Value<2008 and Fields!ExpMonth.Value<5,
"checkbox_false", "checkbox_true")
```

- Running the report shows us:

Card Type	Card Number	
SuperiorCard	33332664695310	<input checked="" type="checkbox"/>
Distinguish	55552127249722	<input checked="" type="checkbox"/>
ColonialVoice	77778344838353	<input checked="" type="checkbox"/>
ColonialVoice	77774915718248	<input checked="" type="checkbox"/>
Vista	11114404600042	<input type="checkbox"/>
Distinguish	55557132036181	<input checked="" type="checkbox"/>
Distinguish	55553635401028	<input checked="" type="checkbox"/>
SuperiorCard	33336081193101	<input checked="" type="checkbox"/>
Distinguish	55553465625901	<input type="checkbox"/>
SuperiorCard	33332126386493	<input checked="" type="checkbox"/>
SuperiorCard	33335352517363	<input checked="" type="checkbox"/>

- Save the report as **Report014.rdl**.

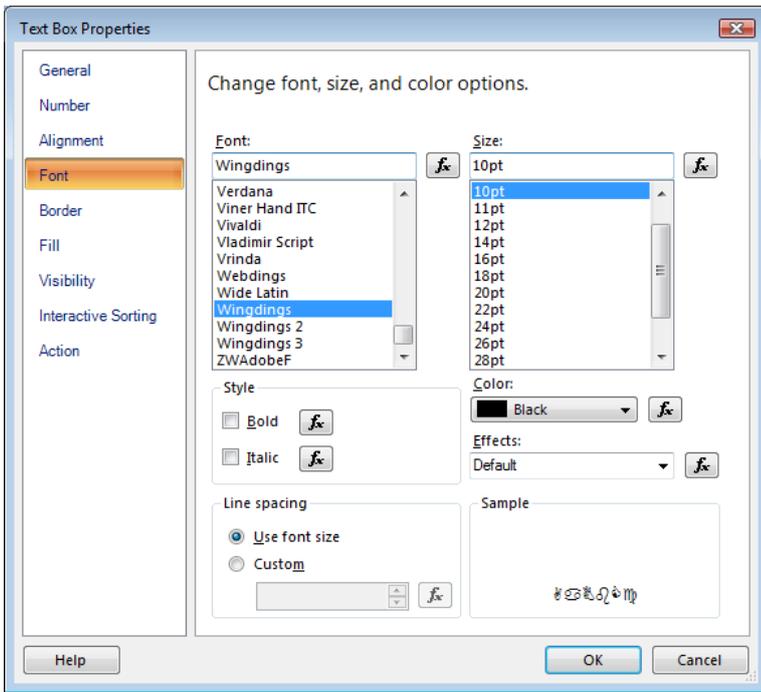
3.1.5.3 Method 2: The Wingdings Font

We'll continue with **Report014.rdl**.

- In our last example we add an additional column to the right.
- Enter the following expression:

```
=IIF(Fields!ExpYear.Value<2008 and Fields!ExpMonth.Value<5, Chr(111), Chr(254))
```

- Change the font to **Wingdings**.



After setting some font-related properties, here's what the rendered report looks like:

Card Type	Card Number		
SuperiorCard	33332664695310	✓	☑
Distinguish	55552127249722	✓	☑
ColonialVoice	77778344838353	✓	☑
ColonialVoice	77774915718248	✓	☑
Vista	11114404600042	□	☐
Distinguish	55557132036181	✓	☑
Distinguish	55553635401028	✓	☑
SuperiorCard	33336081193101	✓	☑
Distinguish	55553465625901	□	☐
SuperiorCard	33332126386493	✓	☑
SuperiorCard	33335352517363	✓	☑

- Save the report (**Report014.rdl**).

A small disadvantage to this method is that the **Wingdings** font needs to be installed on the **SSRS** server.

3.1.6 The Indicator

The **Indicator** is new since **SQL Server 2008 R2!**

- Create a new report.

- Save the report as **Report015.rdl**.
- Data source **dsrAdventureWorks**.
- Dataset **dstOrder**.

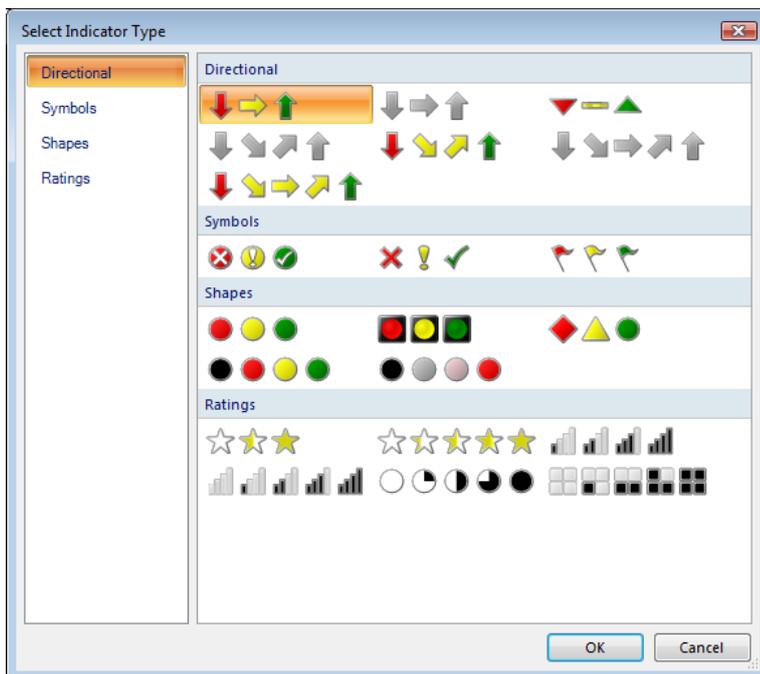
```

SELECT
Sales.SalesOrderHeader.SalesOrderID
,Sales.SalesOrderHeader.TotalDue
FROM
Sales.SalesOrderHeader

```

- Add a table to the blank report
- Add both fields to the table.
- Click the tab **Insert** ⇒ **Indicator**.
- We'll get a pointer which we will drag to the third column.

We then will get:



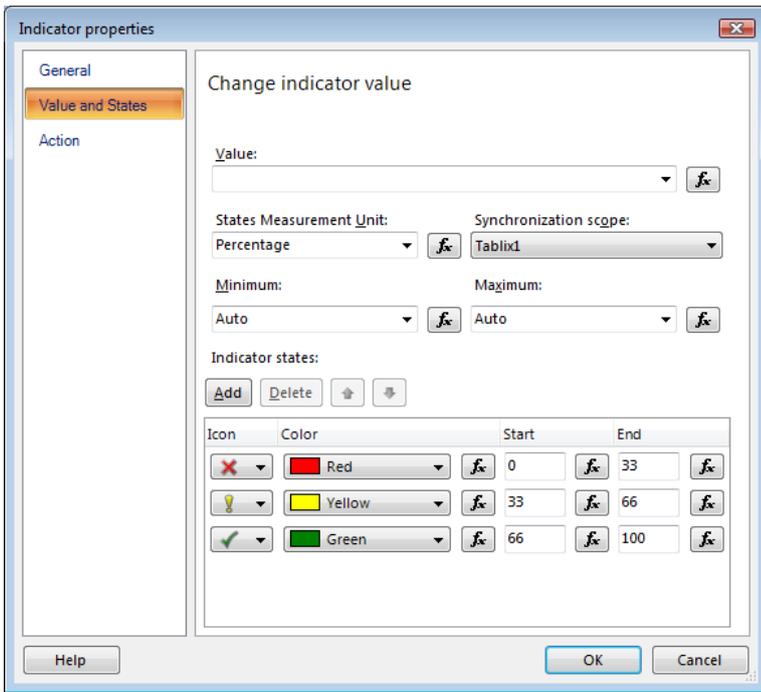
- Click 3 **Symbols** (Uncircled).
- Click **OK**.

Our report will look like this:

Sales Order ID	Total Due	
[SalesOrderID]	[TotalDue]	!

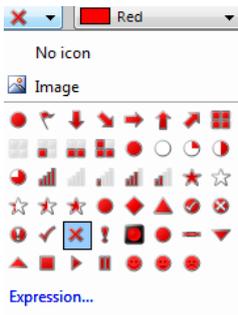
[&ExecutionTime]

- Right click the **exclamation mark**.
- Choose **Indicator Properties**.
- Click **Value and States**.

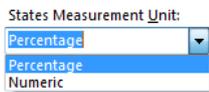


As you can see, the units are measured using percentages with **red** starting at zero while ending at 33. That means that, based on all available values in the dataset, all values in the first 33% will become red.

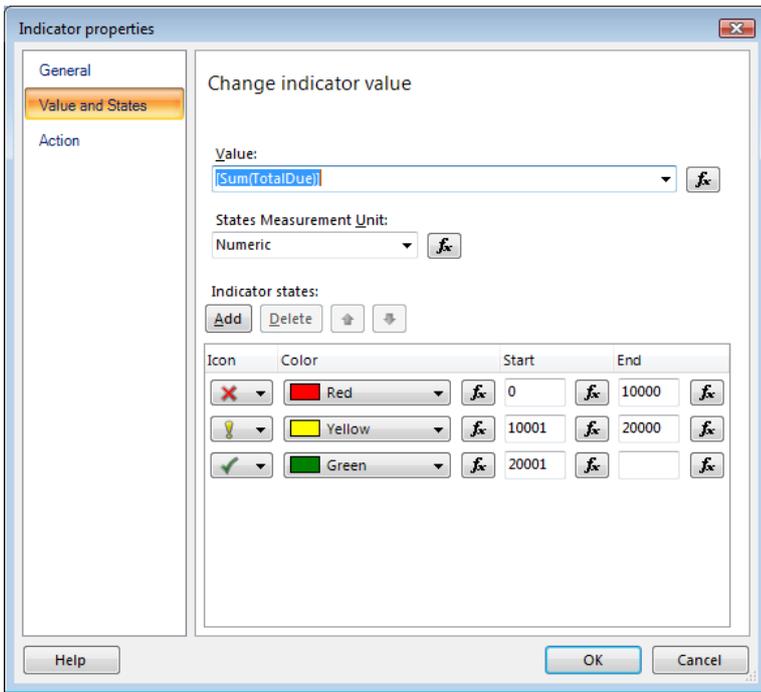
Alternatively, if you'd like to customize the behavior of the indicator, that's an option as well. Have a look at what the **Icon dropdown** produces:



Any of those built-in icons can be selected. Customizing the color is easy too, just use the **Color dropdown**. And of course the numeric ranges can be changed as well. If the percentage-based measurement doesn't work out well in your situation, you can switch to **Numeric**:



- Fill the **Indicator Properties** in like this:



- Run the report.

We'll get:

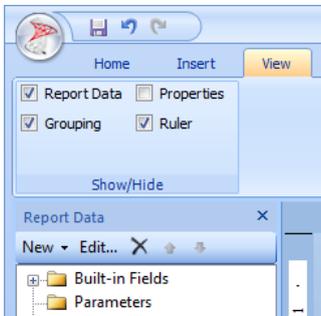
Sales Order ID	Total Due	Indicator State
43659	27231,5495	Green ✓
43660	1716,1794	Red ✗
43661	43561,4424	Green ✓
43662	38331,9613	Green ✓
43663	556,2026	Red ✗
43664	32390,2031	Green ✓
43665	19005,2087	Yellow !
43666	6718,0510	Red ✗
43667	8095,7863	Red ✗
43668	47815,6341	Green ✓
43669	974,0229	Red ✗
43670	8115,6763	Red ✗

- Save the report (**Report015.rdl**).

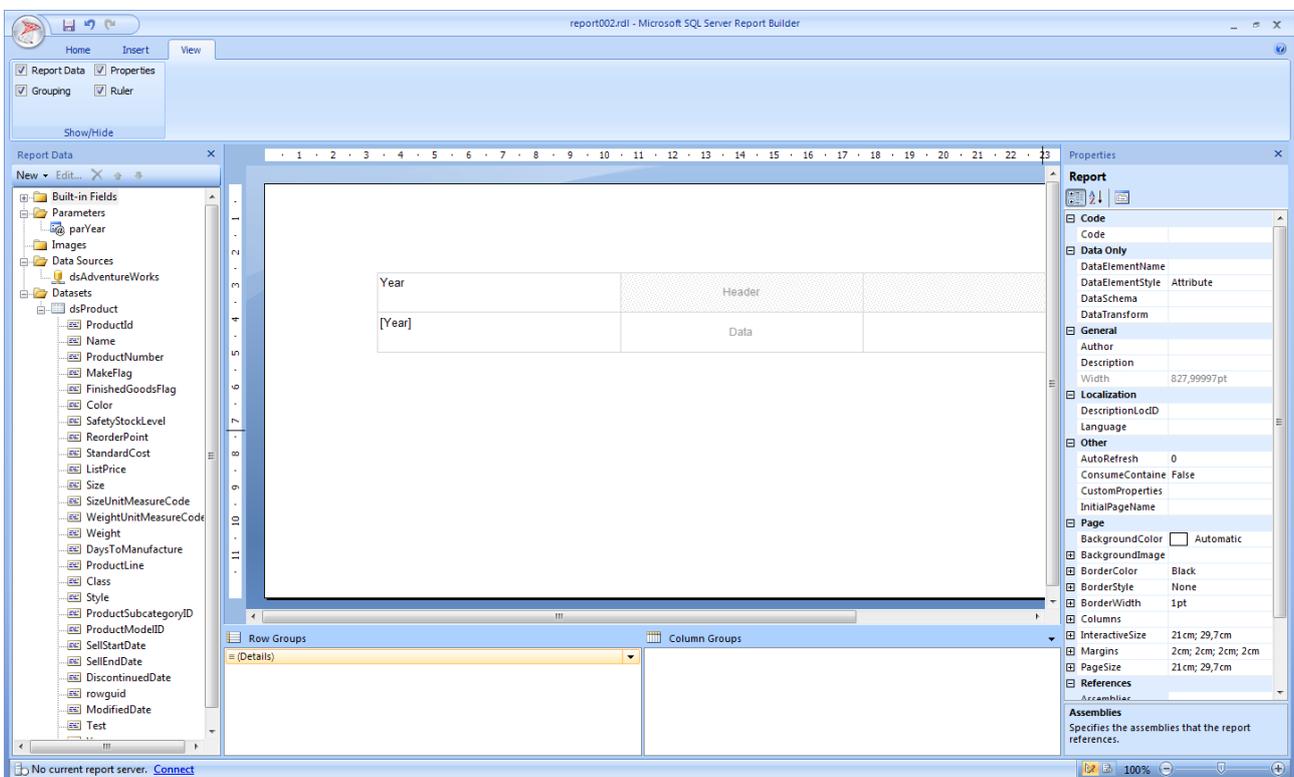
3.2 Page formatting

3.2.1 Margins

- Continue with the last report **Report015.rdl**.
- Click the tab **View**.
- Check **Properties**.

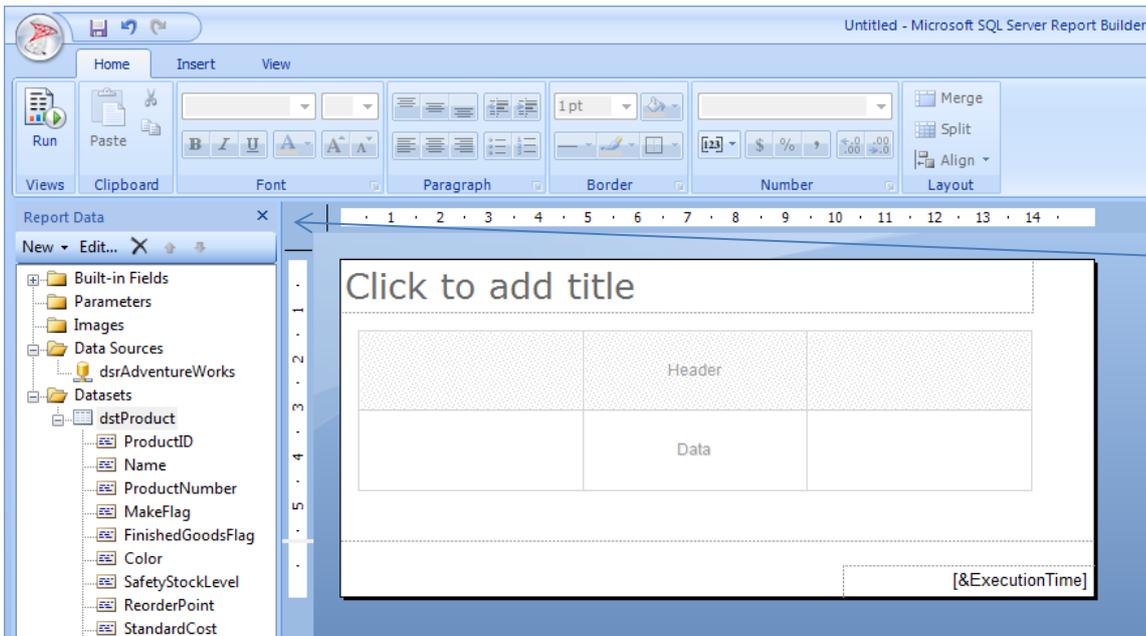


On the right, the properties pane will appear:



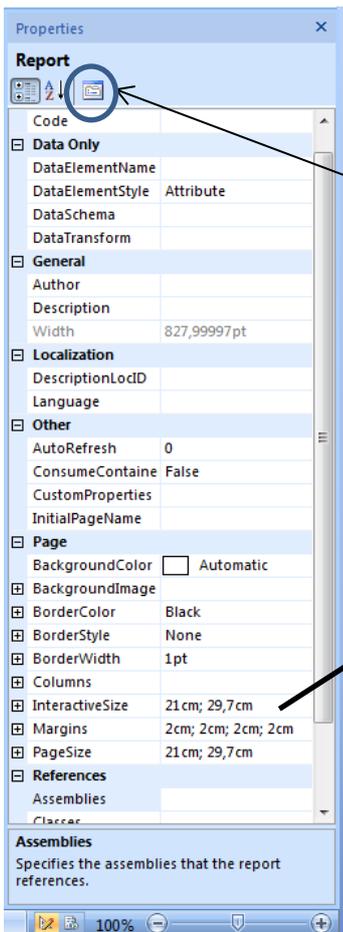
- In order to get the right **Properties** click the upper left corner of the report.

Note: Setting the option **ConsumeContainerWhiteSpace** to true will prevent blank pages at the end.



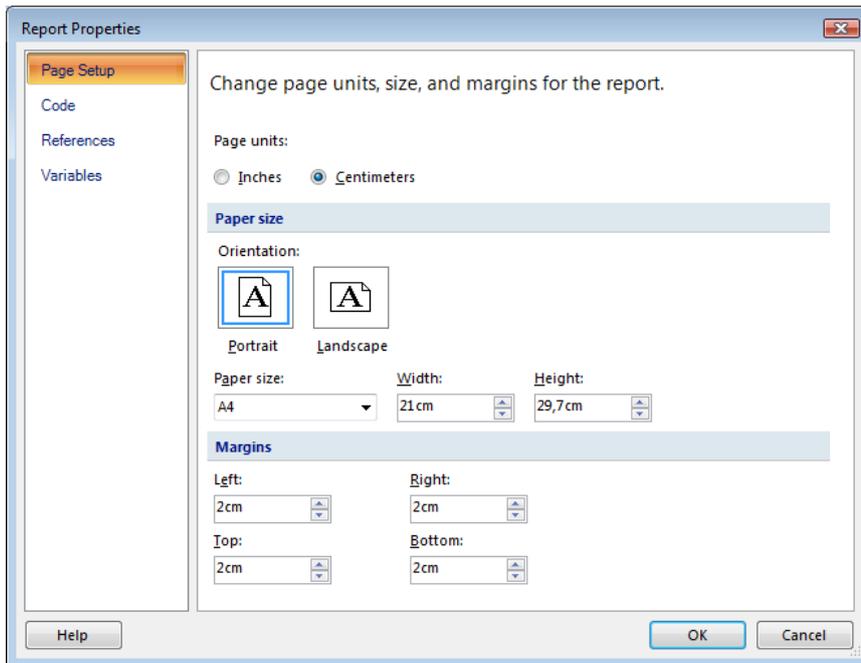
Down below you now can specify the **Page Margins**.

- We can also click the **Property pane** button.



Height = 0; no pages!!!

We will get this pane:



Here we can adjust the margins as well.

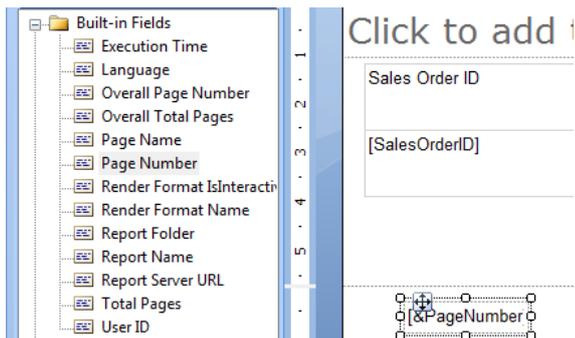
- Leave the margins like they are.
- Save the report (**Report015.rdl**).

3.2.2 Page orientation

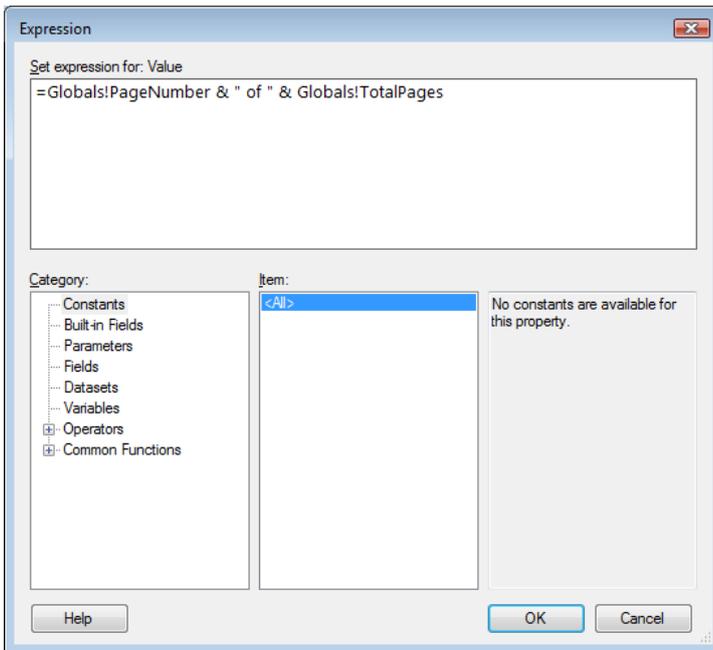
We can specify the **PageSize** in the **Properties** pane as well. In this case we leave it as it is.

3.2.3 Page numbering

- Continue with report **Report015.rdl**.
- We can simply add a page number by dragging the page number field from the **Built-in-fields** to the footer of the report.



- If we want to change the page number into something like **1 of 100** we right click the page number box.
- Choose **Expression**.



- We fill in the formula

=Globals!PageNumber & " of " & Globals!TotalPages

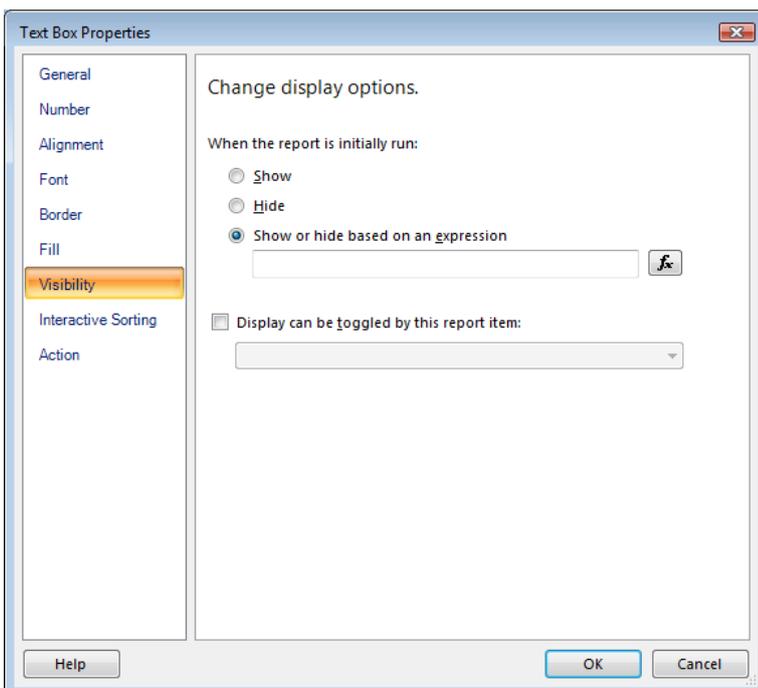
Some more examples.

If we want to start numbering on page 2, we'll need a formula like:

=IIF(Globals!PageNumber = 1, "", cStr(Globals!PageNumber-1))

If we want to suppress the page number on the last page:

- Right click the page number box.
- Choose **Text Box Properties**.



- Click **Visibility**.
- Check **Show or hide based on an expression**.

- Fill in this formula:

```
=IIF(Globals!PageNumber = Globals!TotalPages, False, True)
```

For the next items we need an example.

- Create a new report.
- Save the report as **Report016.rdl**.
- Data source **dsrAdventureWorks**.
- Dataset **dstOrder**.

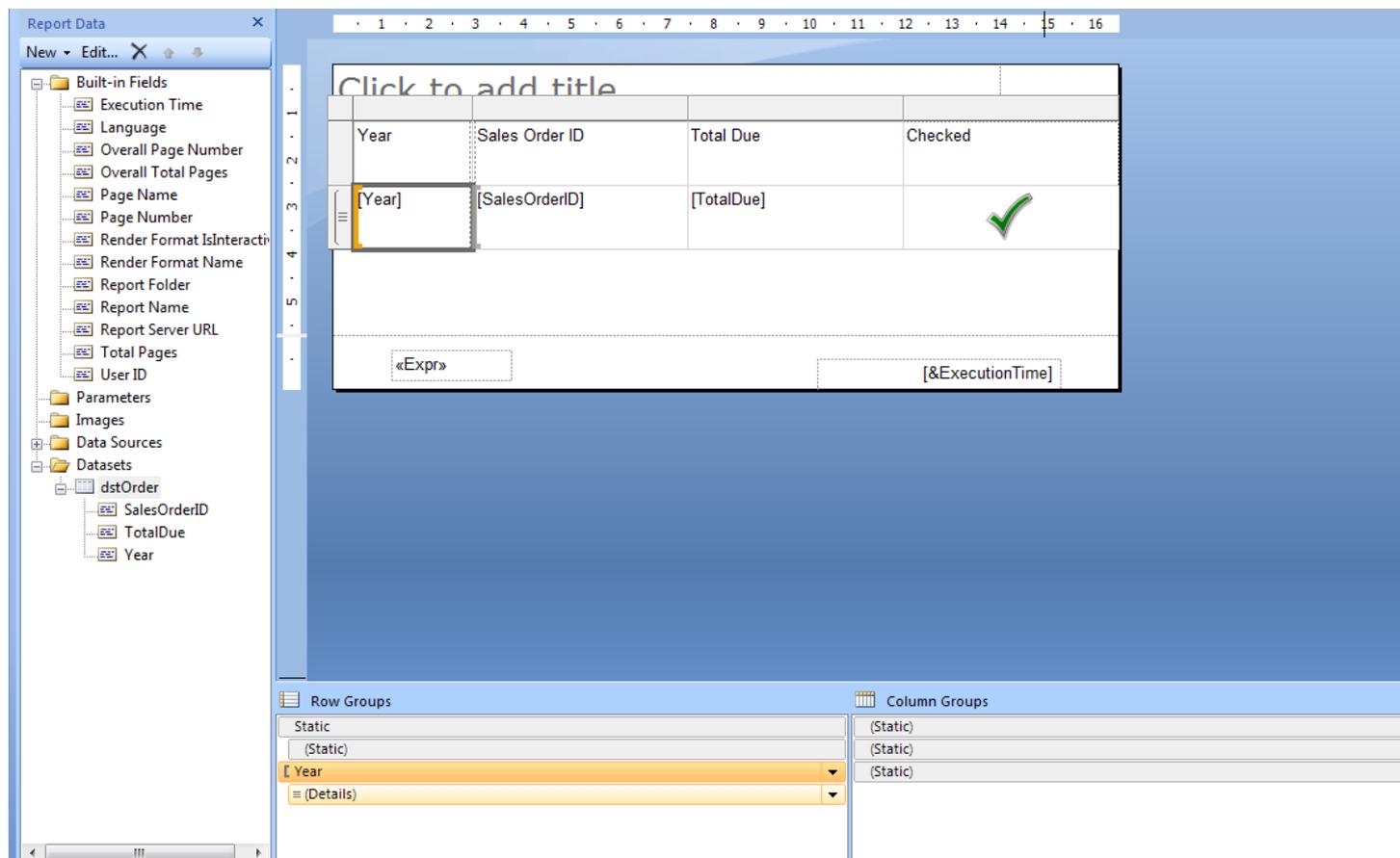
```
SELECT
Sales.SalesOrderHeader.SalesOrderID
, Sales.SalesOrderHeader.TotalDue
, YEAR(Sales.SalesOrderHeader.OrderDate) AS [Year]
FROM
Sales.SalesOrderHeader
```

- Add a table to the blank report
- Add both fields **SalesOrderID** and **TotalDue** to the table.
- Group by **Year**
- Add a **Page Number** to the footer.
- Switch to the Advanced Mode.



- Click **Year** in **Row Groups**.

We will get this:



Reset page numbers with each new page break (e.g. on a group) using the **ResetPageNumber** property, and reference related page counts (Page Number, Total Pages) using

■ **=Globals!PageNumber**

and

■ **=Globals!TotalPages.**

- In the Properties pane at Page Break choose Break location: Between
ResetPageNumber: True
- At Page Name fill in the formula:

■ **=Fields!Year.Value**

You can set the **Page Name** property on a group and refer to its current value from the page header by referencing

■ **=Globals!PageName**

Reference overall page number (**OverallPageNumber**) and total page count (**OverallTotalPages**), unaffected by page number resets, using

■ **=Globals!OverallPageNumber**

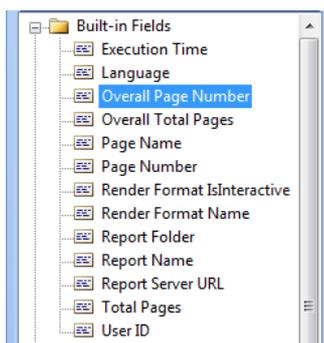
and

■ **=Globals!OverallTotalPages.**

- Add those **Globals** to the report.
- Run the report.
- Check the last page for the number.
- Save the report (**Report016.rdl**).

3.2.4 Built-in fields

By clicking the **Built-in Fields** in the left upper corner we get an overview of those fields.



Using the globals collection

The Globals collection contains the global variables for the report. On the design surface, these variables appear prefixed by an & (ampersand), for example, [&ReportName]. The following table describes the members of the Globals collection.

Member	Type	Description
ExecutionTime	DateTime	The date and time that the report began to run.
PageNumber	Integer	The current page number relative to page breaks that reset the page number. At the beginning of report processing, the initial value is set to 1. The page number increments for each rendered page. To number pages within page breaks for a rectangle, a data region, a data region group, or a map, on the PageBreak property, set theResetPageNumber property to True. Not supported on tablix column hierarchy groups. PageNumber can only be used in an expression in a page header or page footer.
ReportFolder	String	The full path to the folder containing the report. This does not include the report server URL.
ReportName	String	The name of the report as it is stored in the report server database.
ReportServerUrl	String	The URL of the report server on which the report is being run.
TotalPages	Integer	The total number of pages relative to page breaks that reset PageNumber. If no page breaks are set, this value is the same asOverallTotalPages. TotalPages can only be used in an expression in a page header or page footer.
PageName	String	The name of the page. At the beginning of report processing, the initial value is set from InitialPageName, a report property. As each report item is processed, this value is replaced by the corresponding value of PageName from a rectangle, a data region, a data region group, or a map. Not supported on tablix column hierarchy groups. PageName can only be used in an expression in a page header or page footer.
OverallPageNumber	Integer	The page number of the current page for the entire report. This value is not affected by ResetPageNumber. OverallPageNumber can only be used in an expression in a page header or page footer.
OverallTotalPages	Integer	The total number pages for the entire report. This value is not affected by ResetPageNumber. OverallTotalPages can only be used in an expression in a page header or page footer.
RenderFormat	RenderFormat	Information about the current rendering request. For more information, see "RenderFormat" in the next section.

Examples

The following examples show how to use a reference to the **Globals** collection in an expression:

- This expression, placed in a text box in the footer of a report, provides the page number and total pages in the report:

```
=Globals.PageNumber & " of " & Globals.TotalPages
```

- This expression provides the name of the report and the time it was run.

```
=Globals.ReportName & ", dated " & Format(Globals.ExecutionTime, "d")
```

The **User** collection contains data about the user who is running the report. You can use this collection to filter the data that appears in a report, for example, showing only the data of the current user, or to display the UserID, for example, in a report title. On the design surface, these variables appear prefixed by an & (ampersand), for example, [&UserID].

The following table describes the members of the **User** collection.

Member	Type	Description
Language	String	The language of the user running the report. For example, en-US.
UserID	String	The ID of the user running the report. If you are using Windows Authentication, this value is the domain account of the current user. The value is determined by the Reporting Services security extension, which can use Windows Authentication or custom authentication.

3.3 Add a Boolean Parameter for Conditional Visibility

We'll continue with **Report016.rdl**.

To add a Boolean parameter.

- On the design surface, in the Report Data pane, right-click **Parameters**, and click **Add Parameter**.
- In **Name**, type ShowSelections.
- In **Prompt**, type Show selections?
- In **Data type**, from the drop-down list, click **Boolean**.
- Click **Default Values**.
- Click **Specify value**, and then click **Add**.
- In **Value**, type **False**.
- Click **OK**.

To set visibility based on a Boolean parameter

- On the design surface, right-click the text box in the page footer that displays the parameter values, and then click **Text Box Properties**.
- Click **Visibility**.
- Select the option **Show or hide based on an expression**, and then click the expression button **Fx**.
- Type the following expression:

=Not Parameters!ShowSelections.Value

- The text box Visibility option is controlled by the property **Hidden**. Apply the **Not** operator so that when the parameter is selected, the **Hidden** property is false, and the text box will be displayed.
- Click **OK**.
- Click **OK**.
- Preview the report.

The text box that displays the parameter choices does not appear.

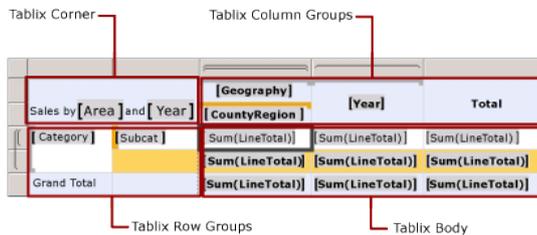
- In the report viewer toolbar, next to **Show selections**, click **True**.
- Preview the report.

The text box in the page footer displays all the store names that you selected.

4 Adding Groups

In **Report Builder**, a group is a named set of data from the report dataset that is bound to a data region. Basically, a group organizes a view of a report dataset. All groups in a data region specify different views of the same report dataset.

To help visualize what a group is, refer to the following figure that shows the tablix data region in Preview. In this figure, the row groups categorize the dataset by product type and the column groups categorize the dataset by geographic region and year.



The following sections help describe the various aspects of groups.

4.1 What Makes a Group?

A group has a name and a set of group expressions that you specify. The set of group expressions can be a single dataset field reference or a combination of multiple expressions. At runtime, group expressions are combined, if the group has multiple expressions, and applied to data in a group. For example, you have a group that uses a date field to organize the data in the data region. At run time, data is organized by date, and then displayed with totals other dataset values for each date.

4.2 When Do We Create Groups?

In most cases, **Report Builder** automatically create a group for you when you design a data region. For a table, matrix, or list, groups are created when you drop fields on the Grouping pane. For a chart, groups are created when you drop fields on the chart drop-zones. For a gauge, you must use the gauge properties dialog box. For a table, matrix, or list, you can also create a group manually.

4.3 How Can We Modify a Group?

After you create a group, you can set data region-specific properties, such as filter and sort expressions, page breaks, and group variables to hold scope-specific data.

To modify an existing group, open the appropriate group properties dialog box. You can change the name of the group. Also, you can specify group expressions based on a single field or multiple fields, or on a report parameter that specifies a value at run time. You can also base a group on a set of expressions, such as the set of expressions that specify age ranges for demographic data.

Note: If you change the name of a group, you must manually update any group expressions that refer to the previous name of the group.

4.4 How are Groups Organized?

Understanding group organization can help you design data regions that display different views of the same data by specifying identical group expressions.

Groups are internally organized as members of one or more hierarchies for each data region. A group hierarchy has parent/child groups that are nested and can have adjacent groups.

If you think of the parent/child groups as a tree structure, each group hierarchy is forest of tree structures. A tablix data region includes a row group hierarchy and a column group hierarchy. Data associated with row group members expands horizontally across the page and data associated with column group members expands vertically down the page. The Grouping pane displays row group and column group members for the currently selected tablix data region on the design surface. For more information, see Grouping Pane.

A chart data region includes a category group hierarchy and a series group hierarchy. Category group members are displayed on the category axis and series group members are displayed on the series axis.

Although typically not needed for gauge data regions, groups do let you specify how to group data to aggregate on the gauge.

4.5 What Types of Groups are Available per Data Region?

Data regions that expand as a grid support different groups than data regions that display summary data visually. Thus, a tablix data region, and the tables, lists, and matrices that are based on the tablix data region, support different groups than a chart or gauge. The following sections discuss the type of and purpose for grouping in each type of data region.

Note: Although groups have different names in different data regions, the principles behind how you create and use groups are the same. When you create a group for a data region, you specify a way to organize the detail data from the dataset that is linked to the data region. Each data region supports a group structure on which to display grouped data.

4.6 Groups in a Tablix Data Region: Details, Row, and Column Groups

As shown earlier in this topic, a tablix data region enables you to organize data into groups by rows or columns. However, row and column groups are not the only groups available in a tablix data region. This data region can have the following types of groups:

- **Details Group**
The Details group consists of all data from a report dataset after **Report Builder** applies dataset and data region filters. Thus, the Details group is the only group that has no group expression.
- Basically, the details group specifies the data that you would see when you run a dataset query in a query designer. For example, you have a query that retrieves all columns from a sales order table. Thus, the data in this detail group includes all the values for every row for all the columns in the table. The data in this detail group also includes values for any calculated dataset fields that you have created.

Note: The data in a Detail group can also include server aggregates, which are aggregates that are calculated on the data source and retrieved in your query. By default, Report Builder treats server aggregates as detail data unless your report includes an expression that uses the **Aggregate** function.

- By default, when you add a table or list to your report, Report Builder and Report Designer automatically create the Details group for you, and adds a row to display the detail data. By default, when you add dataset fields to cells in this row, you see simple expressions for the fields, for example, [Sales]. When you view the data region, the details row repeats once for every value in the result set.
- **Row groups and column groups**
You can organize data into groups by rows or columns. Row groups expand vertically on a page. Column groups expand horizontally on a page. Groups can be nested, for example, group first by [Year], then by [Quarter], then by [Month]. Groups can also be adjacent, for example, group on [Territory] and independently on [ProductCategory].
- When you create a group for a data region, Report Builder and Report Designer automatically add rows or columns to the data region and use these rows or columns to display group data.
- **Recursive hierarchy groups**
A recursive hierarchy group organizes data from a single report dataset that includes multiple levels. For example, a recursive hierarchy group could display an organization hierarchy, for example, [Employee] that reports to [Employee]. Reporting Services provides group properties and built-in functions to enable you to create groups for this kind of report data. For more information, see Creating Recursive Hierarchy Groups (Report Builder and SSRS).

The following list summarizes the way you work with groups for each data region:

- **Table**
Define nested row groups, adjacent row groups, and recursive hierarchy row groups (such as for an organizational chart). By default, a table includes a details group. Add groups by dragging dataset fields to the Grouping pane for a selected table.
- **Matrix**
Define nested row and column groups, and adjacent row and column groups. Add groups by dragging dataset fields to the Grouping pane for a selected matrix.
- **List**
By default, supports the details group. Typical use is to support one level of grouping. Add groups by dragging dataset fields to the Grouping pane for a selected list.

After you add a group, the row and column handles of the data region change to reflect group membership. When you delete a group, you have the choice between deleting the group definition only or deleting the group and all its associated rows and columns.

To limit the data to display or use in calculations for detail or group data, set filters on the group. For more information, see *Add Dataset Filters*, *Data Region Filters*, and *Group Filters (Report Builder and SSRS)*. By default, when you create a group, the sort expression for the group is the same as the group expression. To change the sort order, change the sort expression.

4.7 Understanding Group Membership for Tablix Cells

Cells in a row or column of a tablix data region can belong to multiple row and column groups. When you define an expression in the text box of a cell that uses an aggregate function, for example:

```
=Sum(Fields!FieldName.Value)
```

The default group scope for a cell is the inner most child group to which it belongs. When a cell belongs to both row and column groups, the scope is both innermost groups. You can also write expressions that calculate aggregate subtotals scoped to a group relative to another set of data. For example, you can calculate the percent of a group relative to the column group or to all data for the data region, such as:

```
=Sum(Fields!FieldName.Value) / Sum(Fields!FieldName.Value, "ColumnGroup")
```

4.8 Grouping options

When it comes to grouping, Report Builder offers a lot of possibilities. It is pretty hard to imagine what each possibility would look like. Here we will give some examples to feed your imagination.

- Create a new report.
- Save the report as **Report017.rdl**.
- Remove **Page Footer**.
- Remove **Add Title**.
- Data source **dsrAdventureWorks**.
- Dataset **dstSalesOrder**:

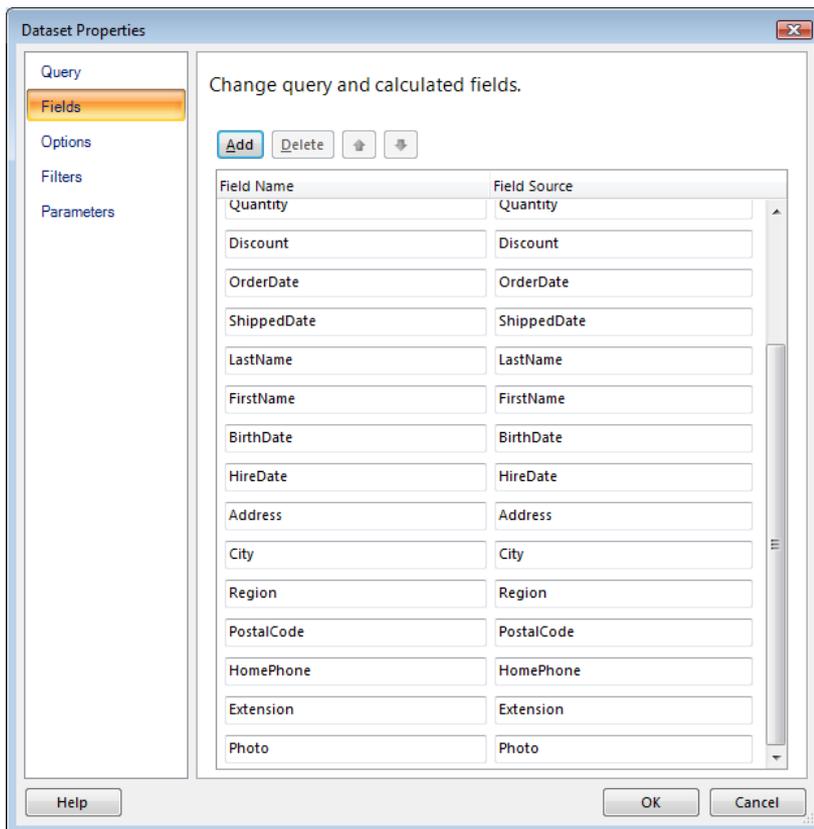
SELECT

```
Categories.CategoryName, OrderDetails.UnitPrice, OrderDetails.Quantity
, OrderDetails.Discount, Products.ProductID, Products.CategoryID
, Orders.OrderDate, Orders.ShippedDate, Employees.LastName
, Employees.FirstName, Employees.HireDate, Employees.BirthDate
, Employees.Photo, Employees.Address, Employees.City
, Employees.Extension, Employees.PostalCode, Employees.HomePhone
, Employees.Region
```

FROM

```
Products INNER JOIN Categories ON Products.CategoryID =
Categories.CategoryID INNER JOIN OrderDetails ON Products.ProductID =
OrderDetails.ProductID INNER JOIN Orders ON OrderDetails.OrderID =
Orders.OrderID INNER JOIN Employees ON Orders.EmployeeID =
Employees.EmployeeID
```

- Choose **Dataset Properties** ⇒ **Fields**.



- Click **Add**.
- Click **Calculated Field**.
- Fill in **Name** in the first box.
- Click **fx** after the last box.
- Fill in:

=YEAR (Fields!OrderDate.Value)

- Click **OK**.
- Add a table to the report.
- Drag the fields **Unit Price**, **Quantity** and **Discount** to the table.

Our example will now look like this:

Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]

It is time to try the various options of grouping.

Column Group: Parent Group.

- Right click [Unit Price].
- Choose **Add Group ⇒ Column Group ⇒ ParentGroup.**



- Group by **Lastname.**

Our example should now look like this:

[Lastname]		
Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]

- Run the report.

Buchanan	Callahan	Davolio	Dodsworth	Fuller	King	Leverling	Peacock	Suyama	Quantity	Discount
Unit Price										
14,0000									12	0
9,8000									10	0
34,8000									5	0
								18,6000	9	0

- Now **Undo** all the actions until you get the starting table again.

Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]

Column Group: Adjacent Left.

- Right click [Unit Price].
- Choose **Add Group ⇒ Column Group ⇒ Adjacent Left.**
- Group by **Lastname.**

Our example should now look like this:

Lastname	Unit Price	Quantity	Discount
[Lastname]	[Unit Price]	[Quantity]	[Discount]

- Run the report.

Last Name	Unit Price	Quantity	Discount								
Buchanan									14,0000	12	0
Buchanan									9,8000	10	0
Buchanan									34,8000	5	0
								Suyama	18,6000	9	0
								Suyama	42,4000	40	0

- Now **Undo** all the actions until you get the starting table again.

Column Group: Adjacent Right.

- Right click [Unit Price].
- Choose **Add Group** ⇒ **Column Group** ⇒ **Adjacent Left**.
- Group by **Lastname**.

Our example should now look like this:

Unit Price	Lastname	Quantity	Discount
[Unit Price]	[Lastname]	[Quantity]	[Discount]

- Run the report.

Unit Price	Last Name	Quantity	Discount								
14,0000	Buchanan									12	0
9,8000	Buchanan									10	0
34,8000	Buchanan									5	0
18,6000									Suyama	9	0
42,4000									Suyama	40	0

- Now **Undo** all the actions until you get the start table again.

Column Group: Parent Group, Child Group.

- Right click [Unit Price].
- Choose **Add Group** ⇒ **Column Group** ⇒ **ParentGroup**.
- Right click [Lastname].
- Group by **Lastname**.
- Choose **Add Group** ⇒ **Column Group** ⇒ **ChildGroup**.
- Group by **Year**.

Our example should now look like this:

[Lastname]		
[YEAR]		
Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]

- Run the report.

Buchanan			Callahan			Davolio			Dedsworth			Fuller		
1996	1997	1998	1996	1997	1998	1996	1997	1998	1996	1997	1998	1996	1997	1998
Unit Price														
14,0000														
9,8000														
34,8000														

- Now **Undo** all the actions until you get the starting table again.

Row Group: Parent Group

- Right click [Unit Price].
- Choose **Add Group** ⇒ **Row Group** ⇒ **ParentGroup**.
- Group by **Lastname**.

Our example should now look like this:

Lastname	Unit Price	Quantity	Discount
[Lastname]	[Unit Price]	[Quantity]	[Discount]

- Run the report.

Last Name	Unit Price	Quantity	Discount
Buchanan	14.0000	12	0
	9.8000	10	0
	34.8000	5	0
	3.6000	15	0,15
	19.2000	21	0,15
	8.0000	21	0
	2.0000	60	0,05
	27.8000	20	0,05
	14.4000	60	0

- Now **Undo** all the actions until you get the starting table again.

Row Group: Adjacent Above.

- Right click **[Unit Price]**.
- Choose **Add Group** ⇒ **Row Group** ⇒ **Adjacent Above**.
- Group by **Lastname**.

Our example should now look like this:

[Lastname]		
Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]

- Run the report.

Buchanan		
Callahan		
Davolio		
Dodsworth		
Fuller		
King		
Leverling		
Peacock		
Suyama		
Unit Price	Quantity	Discount
14.0000	12	0
9.8000	10	0
34.8000	5	0
18.6000	9	0
42.4000	40	0
7.7000	10	0
42.4000	35	0,15

- Now **Undo** all the actions until you get the starting table again.

Row Group: Adjacent Below.

- Right click **[Unit Price]**.
- Choose **Add Group** ⇒ **Row Group** ⇒ **Adjacent Below**.
- Group by **Lastname**.

Our example should now look like this:

Unit Price	Quantity	Discount
[Unit Price]	[Quantity]	[Discount]
[Lastname]		

- Run the report.

32,0000	1	0
18,0000	2	0,05
9,6500	3	0
12,0000	3	0,02
7,0000	2	0
24,0000	2	0
34,0000	2	0,06
33,2500	2	0,03
17,0000	1	0
15,0000	2	0,01
7,7500	4	0
13,0000	2	0
Buchanan		
Callahan		
Davolio		
Dodsworth		
Fuller		
King		
Leverling		
Peacock		
Suyama		

- Now **Undo** all the actions until you get the starting table again.

Row Group: Parent Group, Child Group.

- Right click [**Unit Price**].
- Choose **Add Group** ⇒ **Row Group** ⇒ **ParentGroup**.
- Group by **Lastname**.
- Right click [**Lastname**].
- Choose **Add Group** ⇒ **Row Group** ⇒ **ChildGroup**.
- Group by **Year**.

Our example should now look like this:

[Lastname]	[Year]	Unit Price	Quantity	Discount
		[Unit Price]	[Quantity]	[Discount]

- Run the report.

Last Name	Year	Unit Price	Quantity	Discount
Buchanan	1996	14,0000	12	0
		9,8000	10	0
		34,8000	5	0
		3,6000	15	0,15
		19,2000	21	0,15
		8,0000	21	0
		2,0000	60	0,05
		27,8000	20	0,05
		14,4000	60	0
		27,8000	20	0
		17,2000	30	0

- Now **Undo** all the actions until you get the starting table again.

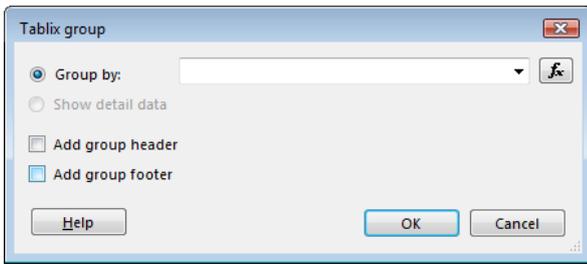
We just showed you two combinations of two groups. There are more. After the first **Row Group** we can choose:

- ChildGroup.**
- Adjacent Below.**
- Adjacent Above.**

And after the first **Column Group**:

- ChildGroup.**
- Adjacent Left.**
- Adjacent Right.**

Within each group, we can choose to add a **group header** and a **group footer**:



But this goes only for the **Parent** and **Child Groups**, not for the other options.

- Save the report (**Report017.rdl**)

5 Calculating Totals

5.1 To group data in a report

- Create a new report.
- Save the report as **Report018.rdl**.
- Create the data source **dsrAdventureWorks**.
- Create the dataset **dstSales**, use the **SQL** query below:

```
SELECT OrderDate AS Date,SalesOrderHeader.SalesOrderID AS [Order],Name AS
Product, OrderQty AS Qty, LineTotal AS [Line Total]
FROM
Sales.SalesOrderHeader
INNER JOIN Sales.SalesOrderDetail
ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID
INNER JOIN Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID
```

- Add a table to the design surface from the **Insert** tab on the ribbon.
- Add the fields **Date**, **Order**, **Name**, **Qty** and **LineTotal** to the table.
- If you do not see the **Row Groups** pane, click the **View** tab.
- Check **Grouping**.
- From the Report Data pane, drag the Date field to the **Row Groups** pane.
- Place it above the row called (Details).

Note that the row handle now has a bracket in it, to show a group. The table now also has two Date columns -- one on either side of a vertical dotted line.

Date	Date	Order
[Date]	[Date]	[Order]

Row Groups

- Date
- (Details)

- From the Report Data pane, drag the **Order** field to the **Row Groups** pane. Place it below **Date** and above (Details).
- Note that the row handle now has two brackets in it, to show two groups. The table now has two **Order** columns, too.
- Delete the original **Date** and **Order** columns to the right of the double line.

This removes this individual record values so that only the group value is displayed.

- Select the column handles for the two columns.
- Right-click.
- Click **Delete Columns**.

Date	Order	Date	Order
[Date]	[Order]	[Date]	[Order]

Row Groups

- Date
- (Details)

- You can format the column headers and date again.
- Switch to the **Home** tab.
- Run the report.

It should look similar to the following illustration:

Date	Order	Product	Qty	Line Total
July 01, 2005	SO43659	AWC Logo Cap	2	\$10.37
		Long-Sleeve Logo Jersey, M	3	\$86.52
		Long-Sleeve Logo Jersey, XL	1	\$28.84
		Mountain Bike Socks, M	6	\$34.20
	SO43661	AWC Logo Cap	4	\$20.75
		Long-Sleeve Logo Jersey, L	4	\$115.36
		Long-Sleeve Logo Jersey, XL	2	\$57.68

5.2 To add totals to a report

- Switch to **Design view**.
- Right-click the data region cell that contains the field **[LineTotal]**
- Click **Add Total**.

This adds a row with a sum of the amount for each order.

- Right-click the cell that contains the field **[Qty]**.
- Click **Add Total**.

This adds a sum of the quantity for each order to the totals row.

- In the empty cell to the left of **Sum[Qty]**, type the label **Order Total**.

You can add a background color to the totals row.

- Select the two sum cells and the label cell.
- On the Format menu, click **Background Color**.
- Click **Light Gray**.
- Click **OK**.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[LineTotal]
	Order Total		[Sum(Qt [Sum(LineTotal)])]	

5.3 To add a daily total to a report

We'll continue working on **Report019.rdl**.

- Right-click the **Order** cell
- Point to **Add Total**.
- Click **After**.

This adds a new row containing sums of the quantity and amount for each day, and the label **Total** in the Order column.

- Type the word **Daily** before the word **Total** in the same cell, so it reads **Daily Total**.
- Select the **Daily Total** cell, the two Sum cells and the empty cell between them.
- On the Format menu, click **Background Color**,
- Click **Orange**.
- Click **OK**.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
		Order Total	[Sum(Qt]	[Sum(LineTotal)
		Daily Total	[Sum(Qt]	[Sum(LineTotal)

5.4 To add a grand total to a report

We'll continue working on **Report018.rdl**.

- Right-click the **Date** cell.
- Point to **Add Total**
- Click **After**.

This adds a new row containing sums of the quantity and amount for the entire report, and the **Total** label in the **Date** column.

- Type the word **Grand** before the word **Total** in the same cell, so it reads **Grand Total**.
- Select the **Grand Total** cell, the two Sum cells and the empty cells between them.
- On the **Format** menu, click **Background Color**.
- Click **Light Blue**.
- Click **OK**.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
		Order Total	[Sum(Qt]	[Sum(LineTotal)
		Daily Total	[Sum(Qt]	[Sum(LineTotal)
Grand Total			[Sum(Qt]	[Sum(LineTotal)

- Click **Run**.

The last page should look something like this:

	S071950	Women's Mountain Shorts, L	6	\$251.96
		Women's Mountain Shorts, S	3	\$125.98
		Order Total	9	\$377.95
	S071951	Women's Mountain Shorts, L	3	\$125.98
		Order Total	3	\$125.98
	S071952	Women's Mountain Shorts, L	15	\$548.55
		Women's Mountain Shorts, M	3	\$125.98
		Women's Mountain Shorts, S	3	\$125.98
		Order Total	21	\$800.51
		Daily Total		2999
Grand Total			84589	\$1,780,769.91

- Save the report (**Report018.rdl**).

6 Calculating Running Totals

6.1 Introduction

We can create a Running total in **Report Builder**. We need the function **RunningValue** to do so. The syntax of this function is like this:

```
=RunningValue(expression, function, scope)
```

- **expression** : The expression on which to perform the aggregation, for example, [Quantity].
- **function** : The name of the aggregate function to apply to the expression, for example- Sum.
- **scope** : The name of a dataset, data region, group or Nothing.

For instance, when we want to create an overall Running total summarizing a field **Unitprice** we would need this formula:

```
=RunningValue(Fields!UnitPrice.Value, SUM, Nothing)
```

Let's create an example.

- Create a new report.
- Save the report as **Report019.rdl**.
- Data source **dsrAdventureWorks**.
- Dataset **dstOrder**:

```
SELECT  
Sales.SalesOrderDetail.SalesOrderID  
,Sales.SalesOrderDetail.OrderQty  
,Sales.SalesOrderDetail.UnitPrice  
,Sales.SalesOrderDetail.LineTotal  
FROM  
Sales.SalesOrderDetail
```

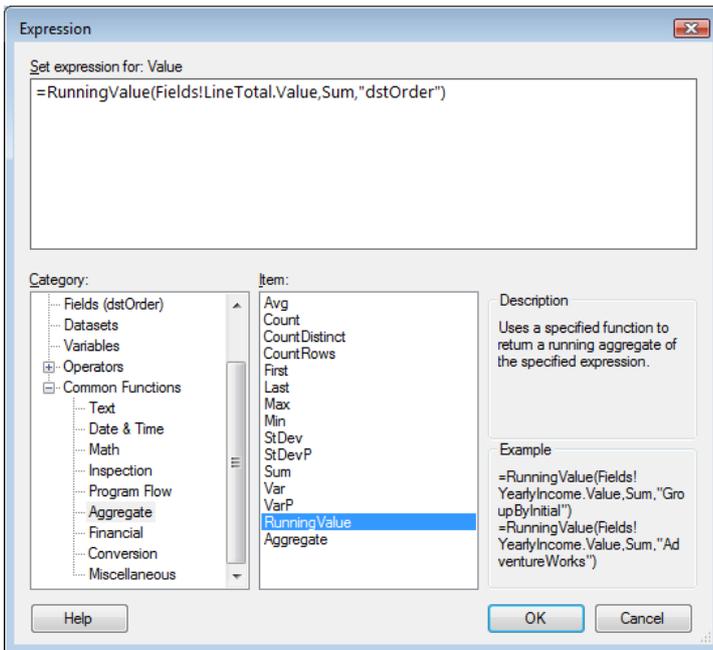
- Add a table to the report.
- Add the columns **SalesOrderID**, **OrderQty**, **UnitPrice** and **Linetotal** to the table.
- Save the report.

Now we will create a Running total.

Instead of adding a field like this to the dataset, we have to add it as an expression directly to the report.

- Add a column to the table to the right.
- Type **Running Total** in the upper cell.
- Right-click the lower cell.
- Click Expression.
- Add the formula:

```
=RunningValue(Fields!LineTotal.Value, SUM, "dstOrder")
```



This means we will get the Running value for all of the dataset.

- Run the report.

Sales Order ID	Order Qty	Unit Price	Line Total	Running Total
43659	1	2024,9940	2024,994000	2024,994000
43659	3	2024,9940	6074,982000	8099,976000
43659	1	2024,9940	2024,994000	10124,970000
43659	1	2039,9940	2039,994000	12164,964000
43659	1	2039,9940	2039,994000	14204,958000
43659	2	2039,9940	4079,988000	18284,946000
43659	1	2039,9940	2039,994000	20324,940000

- Save the report.

6.2 Resetting the RunningValue per Group

Imagine we had a group called **SalesOrderID** then we would use a formula like this:

```
=RunningValue(Fields!LineTotal.Value,Sum,"SalesOrderID")
```

43659	1	2024,994000	2024,994000
	3	6074,982000	8099,976000
	1	2024,994000	10124,970000
	1	2039,994000	12164,964000
	1	2039,994000	14204,958000
	2	4079,988000	18284,946000
	1	2039,994000	20324,940000
	3	86,521200	20411,461200
	1	28,840400	20440,301600
	6	34,200000	20474,501600
	2	10,373000	20484,874600
	4	80,746000	20565,620600
43660	1	419,458900	419,458900
	1	874,794000	1294,252900

6.3 RunningValue in a condition

We can also use the **RunningValue** function in a condition, paragraph 3.1.4: Alternate Row Coloring:

```
=IIF(RunningValue(Fields!CategoryName.Value,COUNTDISTINCT,"Year") MOD 2 =
0,"PaleGreen","White")
```

Here we get a **distinct count** of **CategoryName** within the group **Year**. When the **modulo** division gives a zero, we get the color **PaleGreen**. The result of a **modulo** division is the remainder of an **integer** division.

6.4 Conditional Running Total

This example would only include LineTotals over a 1.000 in the Running Total.

```
=RunningValue(IIf(Fields!LineTotal.Value>1000,Fields!LineTotal.Value,
nothing),Sum,"dstOrder")
```

Sales Order ID	Order Qty	Line Total	
43659	1	2024,994000	2024,994000
43659	3	6074,982000	8099,976000
43659	1	2024,994000	10124,970000
43659	1	2039,994000	12164,964000
43659	1	2039,994000	14204,958000
43659	2	4079,988000	18284,946000
43659	1	2039,994000	20324,940000
43659	3	86,521200	20324,940000
43659	1	28,840400	20324,940000
43659	6	34,200000	20324,940000
43659	2	10,373000	20324,940000
43659	4	80,746000	20324,940000
43660	1	419,458900	20324,940000
43660	1	874,794000	20324,940000
43661	1	809,760000	20324,940000
43661	1	714,704300	20324,940000
43661	2	1429,408600	21754,348600

6.5 Example of conditional, parameterized Running Total

- Create a new report.
- Save the report as **Report020.rdl**.
- Data source **dsrAdventureWorks**.
- Dataset **dstOrder**:

```
SELECT
Sales.SalesOrderDetail.SalesOrderID
,Sales.SalesOrderDetail.LineTotal
FROM
Sales.SalesOrderDetail
```

- Add a table to the report.
- Add the columns **SalesOrderID** and **LineTotal** to the table.
- Create a parameter called **parCondition**, data type **Float**, default value 1000.
- Add a group based on **SalesOrderID**.
- Add a total to the field **LineTotal**
- Save the report.

Now we will create a conditional, parameterized Running total.

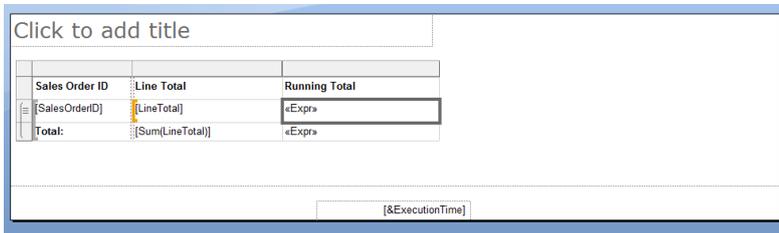
Instead of adding a field like this to the dataset, we have to add it as an expression directly to the report.

- Add a column to the table to the right.
- Type **Running Total** in the upper cell.
- Right-click the lower cell.
- Click **Expression**.
- Add the formula:

```
=RunningValue (iif (Fields!LineTotal.Value > Parameters!parCondition.Value, Fields!LineTotal.Value,nothing) , SUM, "SalesOrderID")
```

- Use the same expression in the Total row.

The design will now look like this:



Click to add title

Sales Order ID	Line Total	Running Total
{SalesOrderID}	{LineTotal}	{Expr}
Total:	{Sum(LineTotal)}	{Expr}

{&ExecutionTime}

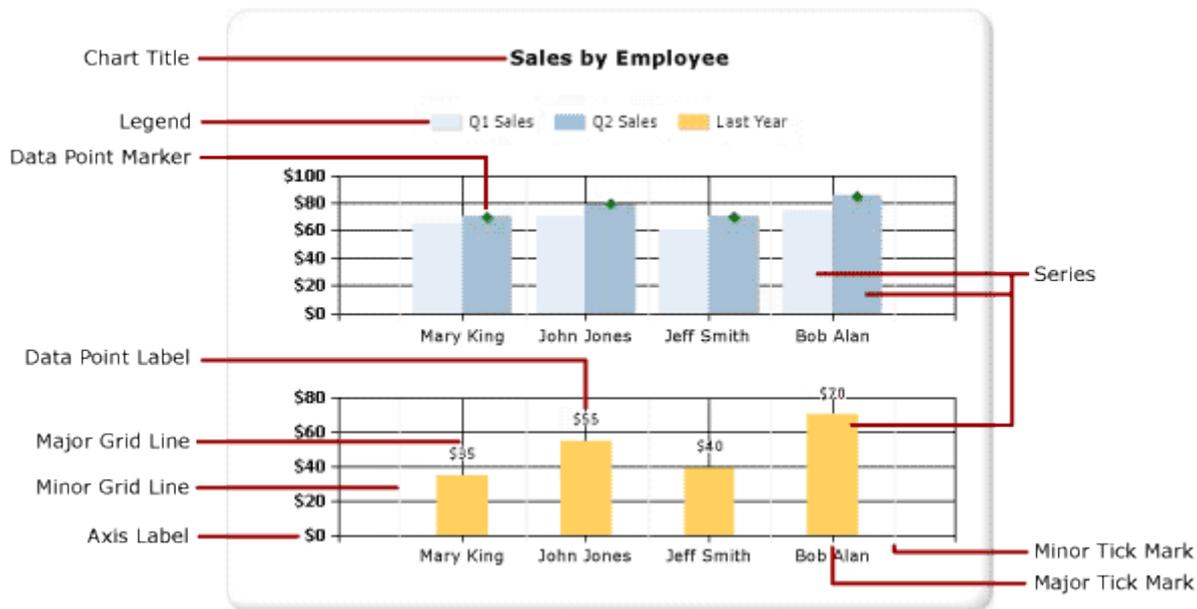
- Run the report.
- Save the report.

7 Adding Graphs

7.1 Introduction

When you want to summarize data in a visual format, use the chart data region. Charts enable you to present large volumes of aggregated information at a glance. It is important to carefully prepare and understand your data before you create a chart, as this will help you design your charts quickly and efficiently.

The following illustration shows many of the different elements used in the chart.



You can publish charts separately from a report as report parts. Report parts are self-contained report items that are stored on the report server and can be included in other reports. Use Report Builder to browse and select parts from the Report Part Gallery to add to your reports. Use Report Designer or Report Builder to save report parts for use in the Report Part Gallery.

7.2 Designing a Chart

After you add a chart data region to the design surface, you can drag report dataset fields for numeric and non-numeric data to the Chart Data pane of the chart. When you click the chart on the design surface, the Chart Data pane appears, with three areas—Category Groups, Series Groups, and Values. If the report has a shared or embedded dataset, the fields in the dataset appear in the Report Data pane. Drag fields from the dataset into the appropriate area. By default, when a field is added to one of the areas of the chart, Reporting Services calculates an aggregate for the field. You can also use series grouping to dynamically generate series. The chart is also closely related to the matrix.

- Create a new report.
- Save the report as **Report021.rdl**.
- Create the data source **dsrAdventureWorks**.
- Create the dataset **dstSales**, use the SQL query below:

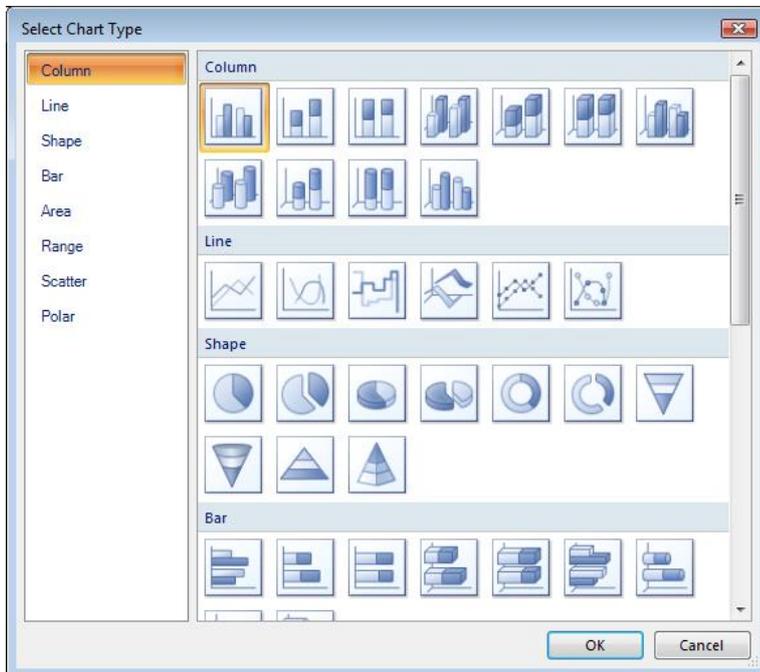
```
SELECT OrderDate AS Date, SalesOrderHeader.SalesOrderID AS [Order], Name AS Product, OrderQty AS Qty, LineTotal AS [Line Total]
FROM
Sales.SalesOrderHeader
INNER JOIN Sales.SalesOrderDetail
ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID
INNER JOIN Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID
```

- Add a Calculated field to the dataset:

=year (Fields!Date.Value)

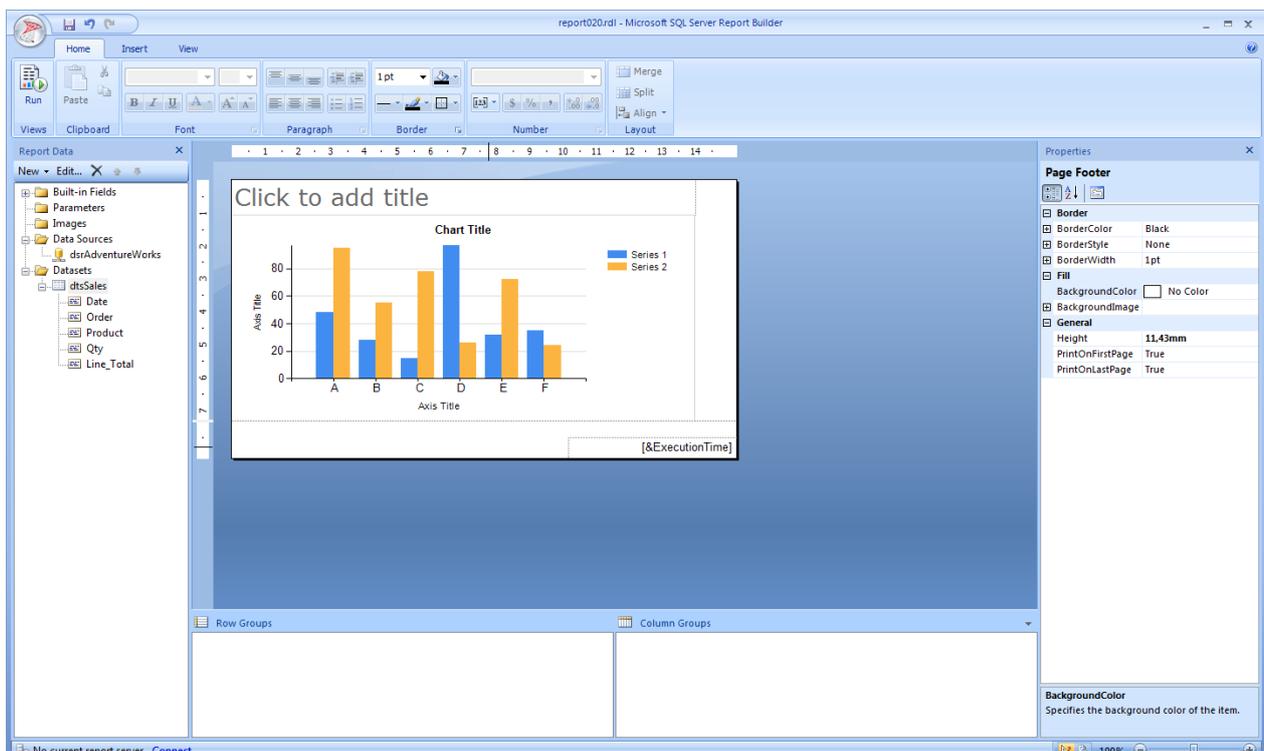
- Go the tab **Insert**.
- Click **Chart** ⇒ **Insert Chart**.
- Click somewhere on the report.

We'll get:



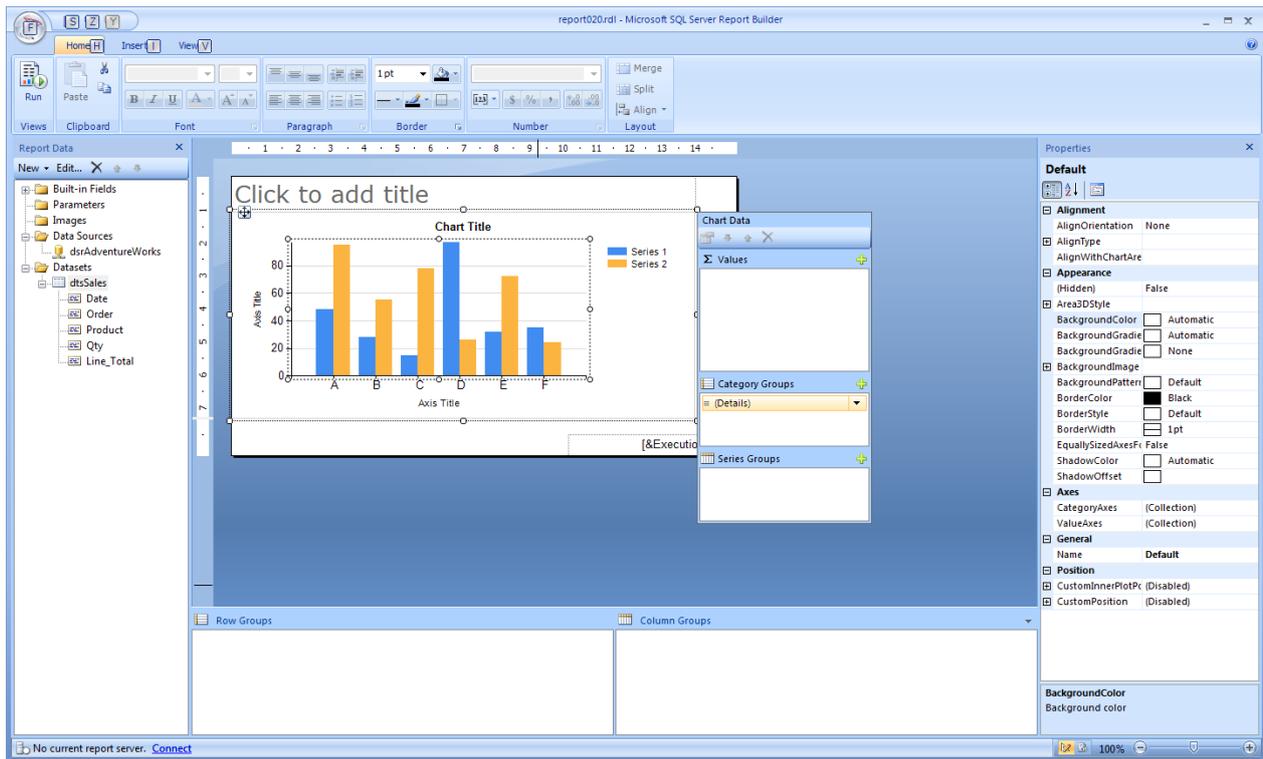
- Click the first **Column** graph.
- Click **OK**.

We'll get:



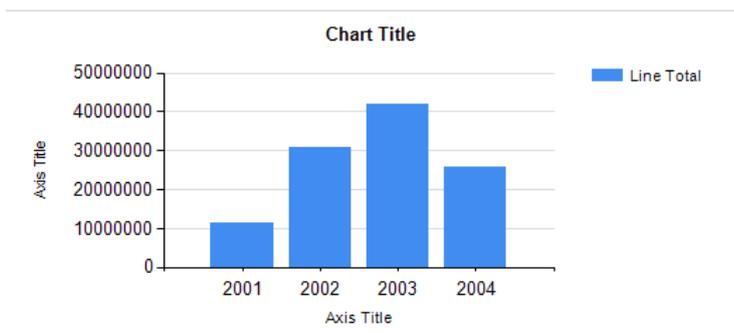
- Click the graph twice.

We'll get:



- Drag the field **Line_Total** to **Values**.
- Add the field **Year** to the **Category Groups**.
- Run the report.

We'll get:



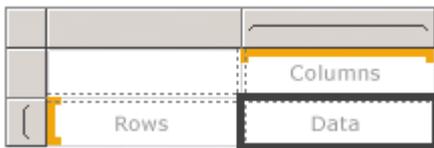
1-10-2013 9:09:27

- Save the report.

Note: The data in the chart at design time is different from the data in the chart when the report is processed. It is not your real data. It is generated data that has been added so that you can design your chart with an idea of what the chart will look like.

7.3 Similarities to a Matrix

One way to think about how charts work is to compare them to matrices.



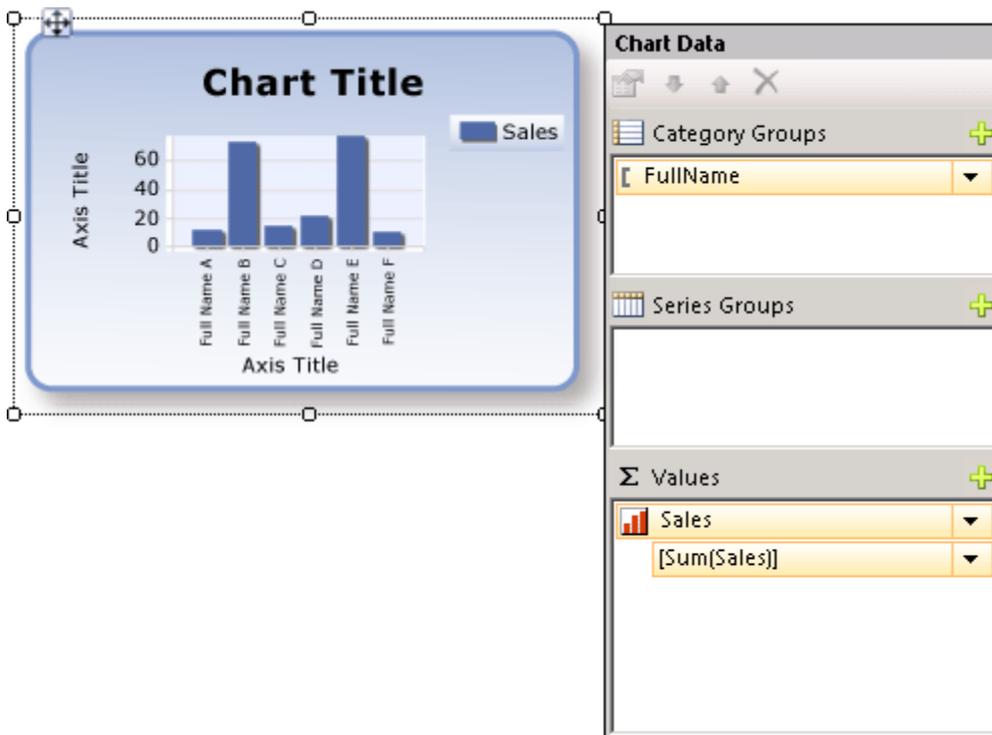
Conceptually, their organization is identical:

- The Columns group in the matrix is identical to the Category Groups area in the chart.
- The Rows group in the matrix is identical to the Series Groups area in the chart.
- The Data area in the matrix is identical to the Values area in the chart.

7.4 Adding Data to the Chart

Suppose you have a report that shows Sales by Name. You drop the Full Name field to the Category Groups area and the Sales field to the Values area.

When you add the Sales field to the Values area, the text of the data field appears in the legend, and the data from this numeric field will be aggregated into one value. By default, the value is aggregated using the built-in function Sum. The Chart Data pane will contain a simple expression for your field. In our example, `[Sum(Sales)]` will appear for the field expression=`Sum(Fields!Sales.Value)`. If no groups are specified, the chart will only show one data point. In order to show multiple data points, you must group your data by adding a grouping field. When you add the Name field to the Category Groups area, a grouping field of the same name as the name of the field is automatically added to the chart. When fields that define the values along the x and y axes are added, the chart has enough information to plot the data correctly.



When the Series Groups area is left empty, the number of series is fixed at design time. In this example, Sales is the only series that appears on the chart.

7.5 Category and Series Groups in a Chart

A chart supports nested category and series groups. Charts do not display detail data. Add groups to a chart by dragging dataset fields to the category and series drop zones for a selected chart.

Shape charts such as pie charts support category groups and nested category groups. Other charts such as bar charts support category groups and series groups. You can nest groups, but make sure that the numbers of categories or series do not obscure the presentation of information in the chart.

Adding Series Grouping to a Chart

If you add a field to the Series Groups area, the number of series depends on the data that is contained in that field. In our earlier example, suppose you add a Year field to the Series Groups area. The number of values in the Year field will determine how many series will appear on the chart. If the Year field contains the years 2004, 2005, and 2006, the chart will display three series for every field in the Values area.

7.6 Dataset Considerations Before Creating a Chart

Charts provide a summary view of your data. However, with large datasets, the information on a chart can become obscured or unreadable. Missing or null data points, data types ill-suited to the type of chart, and advanced applications such as combining charts with tables can all affect the readability of a chart. Before designing a chart, you should carefully prepare and understand your data so that you can design your charts quickly and efficiently.

You can have as many charts in your report as you want. A chart, like any other data region such as a matrix or table, is bound to a single dataset. If you want to display multiple datasets on the same chart, you can create an additional dataset that uses a **JOIN** or **UNION** statement in your **SQL** query before adding data to the chart. For more information about the **JOIN** and **UNION** statement, see Books Online or another **SQL** reference.

Consider pre-aggregating data in the dataset query if detail data is not necessary or useful. To display each data point more clearly, reduce the number of categories in your dataset. You can filter the dataset or add a condition to your query that reduces the number of rows returned.

7.7 Best Practices When Displaying Data in a Chart

Charts are most effective when the number of elements that are displayed presents a clear image of the underlying information. Some charts, like scatter graphs, benefit from numerous data points, while others, like pie charts, are more effective with fewer data points. Choose a chart type carefully based on the values in your dataset and how you want this information to be shown.

There are several ways you can consolidate data on a chart:

- When using a pie chart, collect small slices into one slice called **Other**. This will reduce the number of slices on your pie chart.
- Avoid using data point labels when there are numerous data points. Data point labels are most effective when there are only a few points on the chart.
- Filter unwanted or irrelevant data. This helps you highlight the key data that you are trying to show on the chart. To filter data points in a chart, set a filter on a category group or a series group. By default, the chart uses the built-in function **Sum** to aggregate values that belong to the same group into an individual data point in the series. If you change the aggregate function of a series, you must also change the aggregate function in the filter expression.
- To display ratio data in a table or matrix template, consider using a linear gauge instead of a bar graph. Gauges are better suited for showing a single value inside a cell.

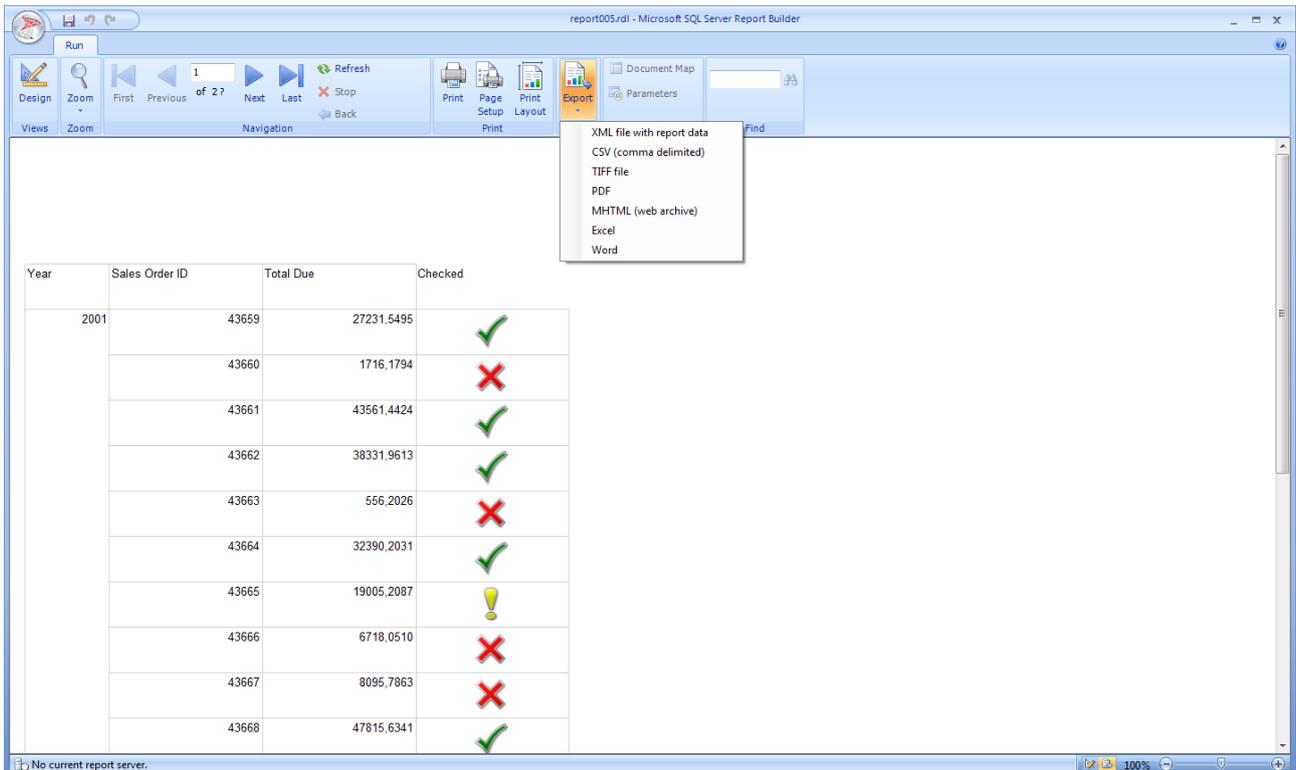
7.8 Aggregating Values from a Data Field on the Chart

By default, when a field is added to the **Values** area of the chart, **Reporting Builder** calculates an aggregate for the field. If you drag a field onto the chart without dropping the field into a specific area, the chart will determine whether this field belongs on the category (x) axis or value (y) axis based on the data type of the field. **Numeric** fields that are dropped in the Values area are aggregated using the **SUM** function. If the data type of your value field is **String** in the Values area, the chart cannot display a numeric value, even if there are numbers in the fields, so the chart displays the **COUNT** function. To avoid this behavior, make sure that the fields that you use have numeric data types, instead of **Strings** that contain formatted numbers. You can use a Visual Basic expression to convert **String** values to a numeric data type using the **Cdbl** or **CInt** constant. For example, the following complex expression converts a field that is named MyField that contains numeric values that are formatted as Strings.

■ =Sum(Cdbl(Fields!MyField.Value))

8 Exporting the Data

To export a report to a different format, you have to run the report.



The screenshot shows the Microsoft SQL Server Report Builder interface. The 'Export' menu is open, displaying the following options:

- XML file with report data
- CSV (comma delimited)
- TIFF file
- PDF
- MHTML (web archive)
- Excel
- Word

The report data is displayed in a table with the following columns: Year, Sales Order ID, Total Due, and Checked. The data is as follows:

Year	Sales Order ID	Total Due	Checked
2001	43659	27231,5495	✓
	43660	1716,1794	✗
	43661	43561,4424	✓
	43662	38331,9613	✓
	43663	556,2026	✗
	43664	32390,2031	✓
	43665	19005,2087	!
	43666	6718,0510	✗
	43667	8095,7863	✗
	43668	47815,6341	✓

9 Complex Exercise

9.1 Information analysis

9.1.1 The information we want

We want a report that gives an overview per *Sales Representative* of the sales per year per category. The sales per year we want divided into columns for gross and net sales, the total reduction as a percentage (read: the reduction per category) and the average supply time per category. We want the gross sales per sales person and per year.

Of every sales Representative we want his/her personal data on the report. Finally the report should have a graph which shows per sales Representative and per year per category the gross sales.

The database

The database we use is: **NorthwindSQL**

Tables

We need the following tables: **Employees, Orders, OrderDetails, Products, Categories.**

Fields

Employees: LastName +FirstName Photo BirthDate HireDate Extension Address City PostalCode HomePhone Region	Orders: OrderDate ShippedDate	OrderDetails: UnitPrice Quantity Discount
Products: ProductID CategoryID	Categories: CategoryName	

Links

Those tables should be linked on the following fields:

- Employees.EmployeeId <-> Orders.EmployeeId
- Orders.OrderId <-> OrderDetails.OrderId
- Order Details.ProductId <-> Products.ProductId
- Products.CategoryId <-> Categories.CategoryId

Derived fields

Some data we cannot find directly in a field, we need to do some transforming:

- Name = LastName & ", " & FirstName
- Gross sales = Order Details.Quantity*Order Details.UnitPrice
- Net sales = Order Details.Quantity*Order Details.UnitPrice * (1-Order Details.Discount)
- Reduction = 1-total net sale per category/total gross sale per category
- Reduction2 = 1-total net sale per year / total gross sale per year
- Supply time = orders.ShippedDate – orders.OrderDate

Filters

Filters are in this case not really necessary. But we could for example filter on sales people by choosing one specific name.

Summaries

We have to do quite some summarizing:

- Total gross and net sales **per category per year per employee**

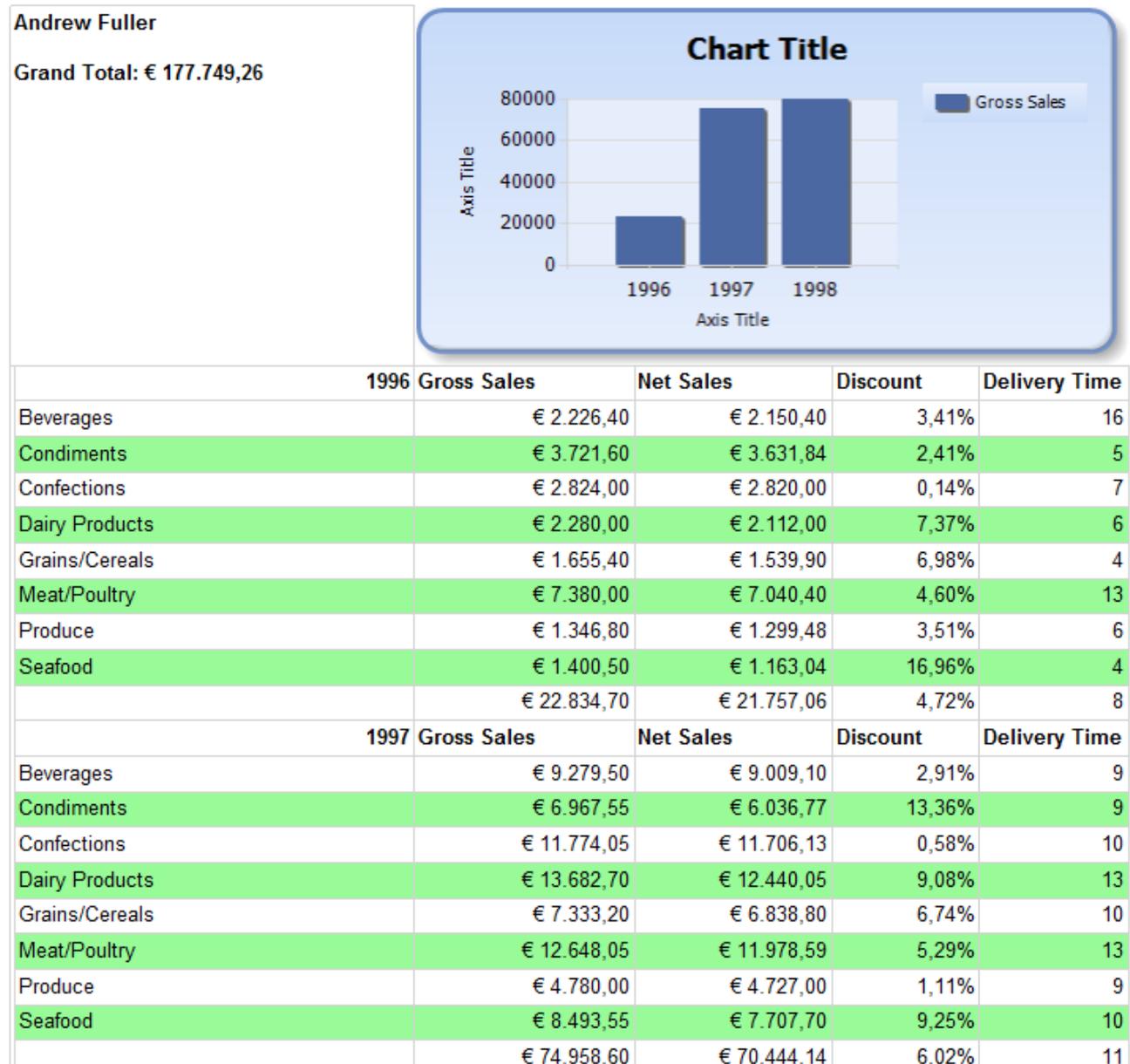
- Average reduction **per** category **per** year **per** employee
- Average supply time **per** category **per** year **per** employee
- Total gross and net sales **per** year
- Total gross sales **per** employee

Sort order

Sorted by employee per year per category

9.1.2 What it should look like

An image of what the report should look like.



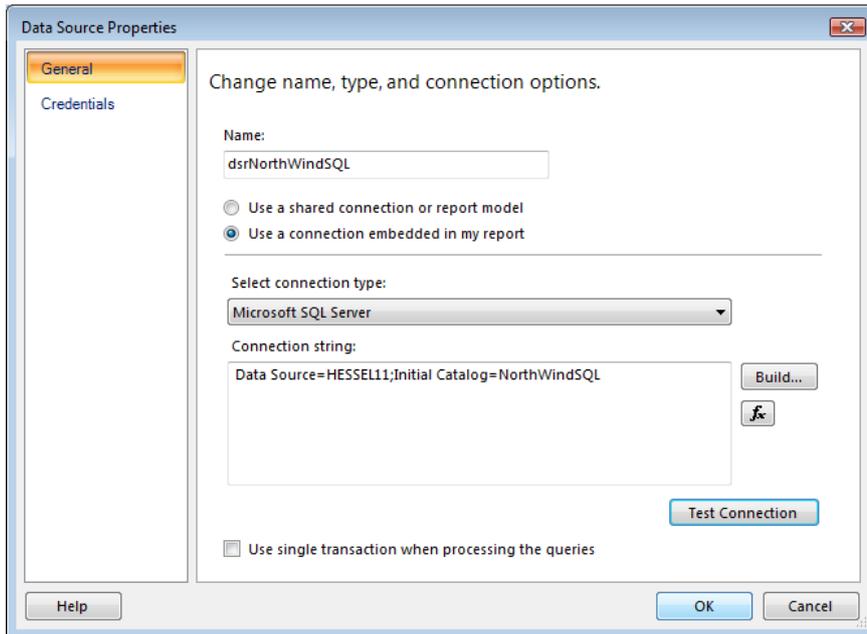
9.2 Creating the report in Report Builder

9.2.1 The structure and the numbers

Of course we start with creating a new report.

- Create a new report.
- Save the report as **report100.rdl**.
- Remove **Page Footer**.
- Remove **Add Title**.

- Data source **dsrNorthWindSQL**.



- Dataset **dstSalesOrder**:

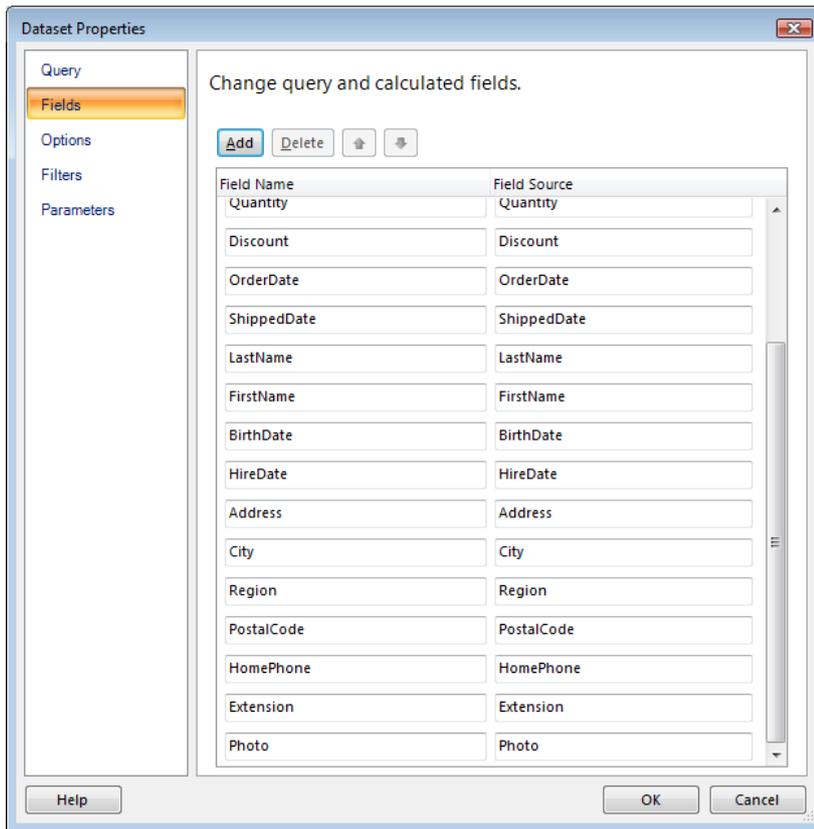
```

SELECT
Categories.CategoryName,OrderDetails.UnitPrice,OrderDetails.Quantity
,OrderDetails.Discount,Products.ProductID,Products.CategoryID
,Orders.OrderDate,Orders.ShippedDate,Employees.LastName
,Employees.FirstName,Employees.HireDate,Employees.BirthDate
,Employees.Photo,Employees.Address,Employees.City
,Employees.Extension,Employees.PostalCode,Employees.HomePhone
,Employees.Region
FROM
Products
INNER JOIN Categories
ON Products.CategoryID = Categories.CategoryID
INNER JOIN OrderDetails
ON Products.ProductID = OrderDetails.ProductID
INNER JOIN Orders
ON OrderDetails.OrderID = Orders.OrderID
INNER JOIN Employees
ON Orders.EmployeeID = Employees.EmployeeID

```

Now we will create some calculated fields.

- Choose Dataset Properties ⇒ Fields.



- Click **Add**.
- Click **Calculated Field**.
- Fill in **Name** in the first box.
- Click **fx** after the last box.
- Fill in:

=Fields!FirstName.Value & " " & Fields!LastName.Value

- Click **OK**.
- Now create the following **Calculated fields GrossSales**

=Fields!Quantity.Value*Fields!UnitPrice.Value

- **NetSales:**

=Fields!Quantity.Value*Fields!UnitPrice.Value*(1-Fields!Discount.Value)

- **SupplyTime:**

=Fields!ShippedDate.Value-Fields!OrderDate.Value

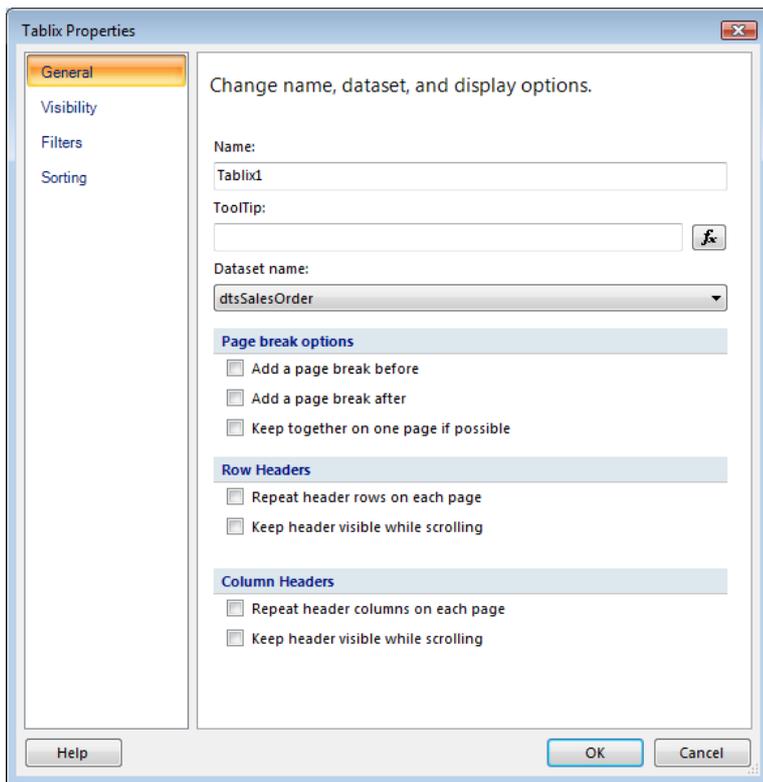
- **Year:**

=YEAR(Fields!OrderDate.Value)

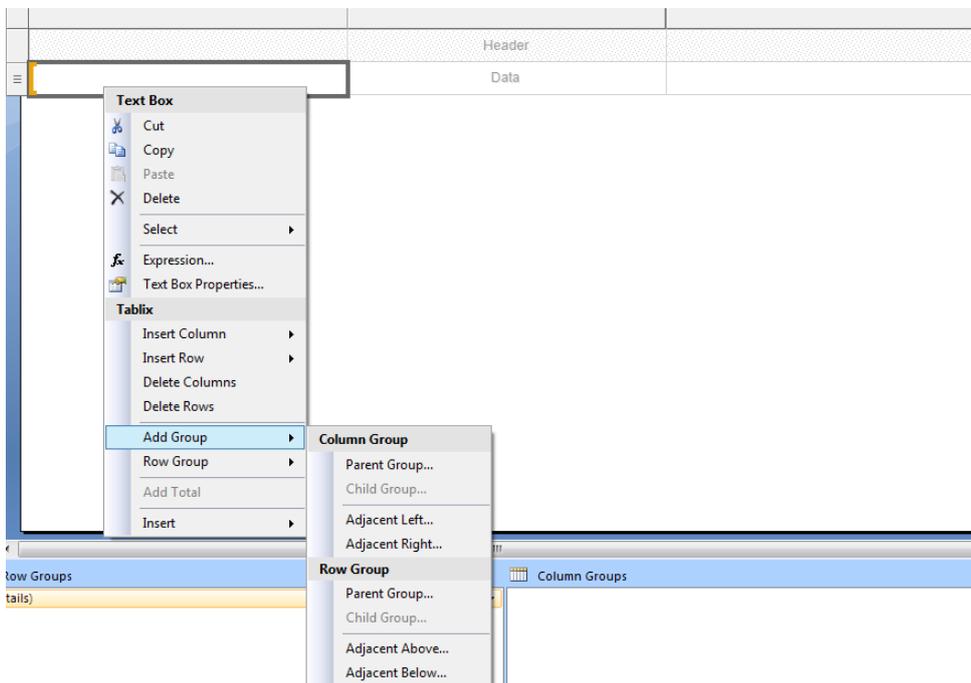
- Add a table to the report.

Now we have to group the data. When you have not added any fields yet to the table, the dataset is not yet linked to the table. In order to create groups, this has to be done first.

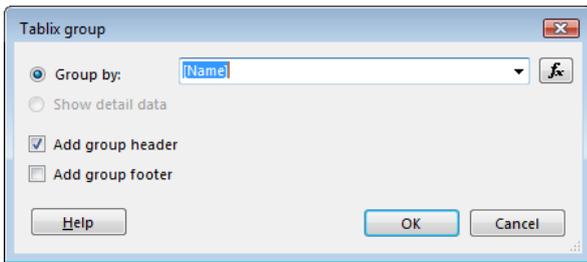
- Right-click the table at the row or column handler.
- Click **Table Properties**.
- Choose **dtsSalesOrder** as **Dataset Name**.



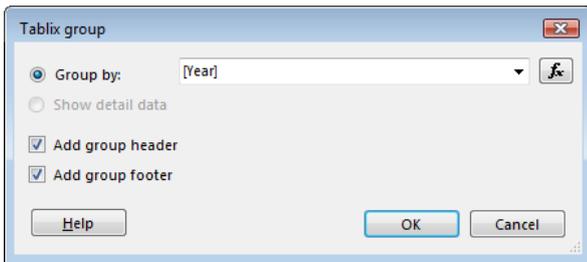
- Right-click the table.
- Click **Add Group**.



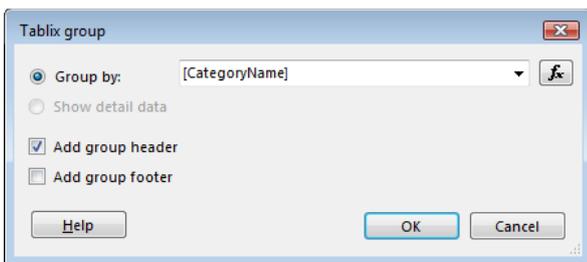
- Click **Row Group** ⇒ **Parent Group**.
- Fill it in like this:



- Click **Row Group** ⇒ **Child Group**.
- Fill it in like this:



- Click **Row Group** ⇒ **Child Group**.
- Fill it in like this:



- Right-click **Details** in the **Row Groups** underneath.
- Click **Group Properties**.
- Click **Visibility**.
- Check **Hide**.
- Click **OK**.

Right now it will look like this:

Name	Year	Category			
[Name]					
	[Year]	[CategoryName]			

Now we want to have **Name**, **Year** and **Category** all in one column.

- Right click the **[Name]** cell.

- Click **Split cells**.
- Delete the label **Year**.
- Now, select the **Name** cell and the cell adjacent right.
- Right-click the selection.
- Choose **Merge Cells**.
- Now, select the [**Name**] cell and the cell adjacent right.
- Right-click the selection.
- Choose **Merge Cells**.
- Right click the [**Year**] cell.
- Click **Split cells**.
- Add CategoryName to the cell below.
- Delete the third column.
- Now make the first column as small as possible.

Take into account that this is all pretty tricky. The result should look like this.

Name			
[Name]			
Year			
[Year]			
[CategoryName]			

- Run the report as a test.

The result should look like this:

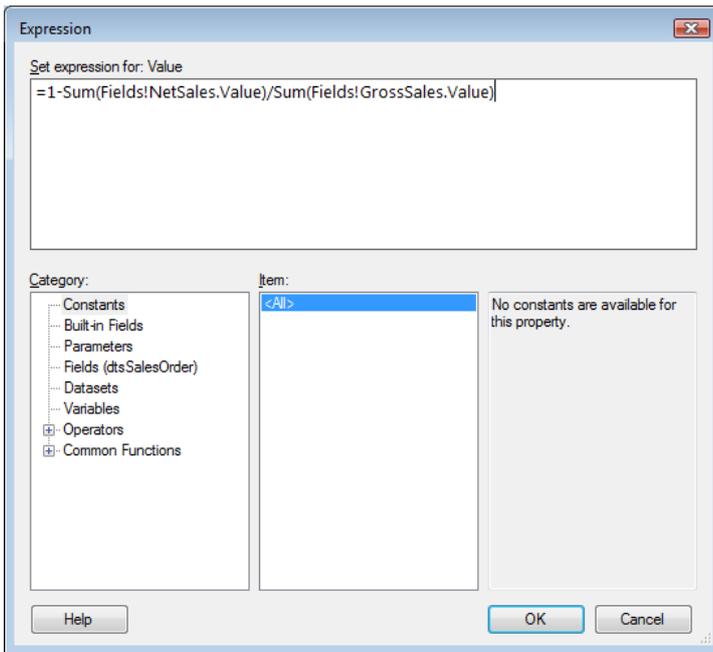
Andrew Fuller	
Year	
1996	
Beverages	
Condiments	
Confections	
Dairy Products	
Grains/Cereals	
Meat/Poultry	
Produce	
Seafood	
Year	
1997	
Beverages	
Condiments	
Confections	
Dairy Products	
Grains/Cereals	
Meat/Poultry	
Produce	
Seafood	

Now we have to add the summaries.

- In the cells just after [**Category**] choose again **GrossSales** and **NetSales**.

The choices will automatically turn into [**SUM(GrossSales)**].

- Right-click the cell just after [**SUM(NetSales)**] one by one.
- Click **Expression**.
- Fill in:



- Repeat this for the footer of [Year].

Now, the average supplying time is left.

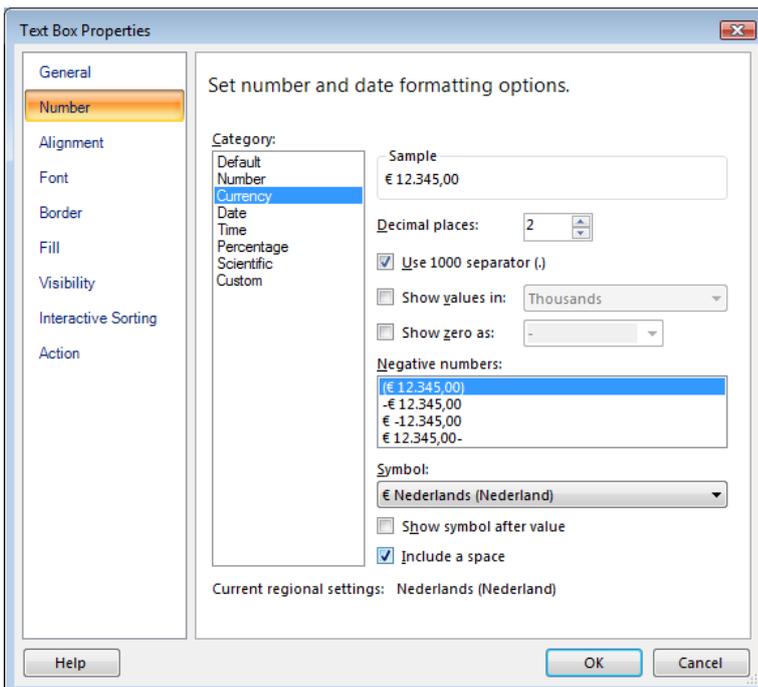
- At the right end of the table add an extra column.
- Just adjacent right to the other expressions, add:
=Avg (DateDiff ("d" , Fields! OrderDate . Value , Fields! ShippedDate . Value))

Now, we have all the numbers. The formatting can start.

9.2.2 Formatting

- Format all the number the way you want them.

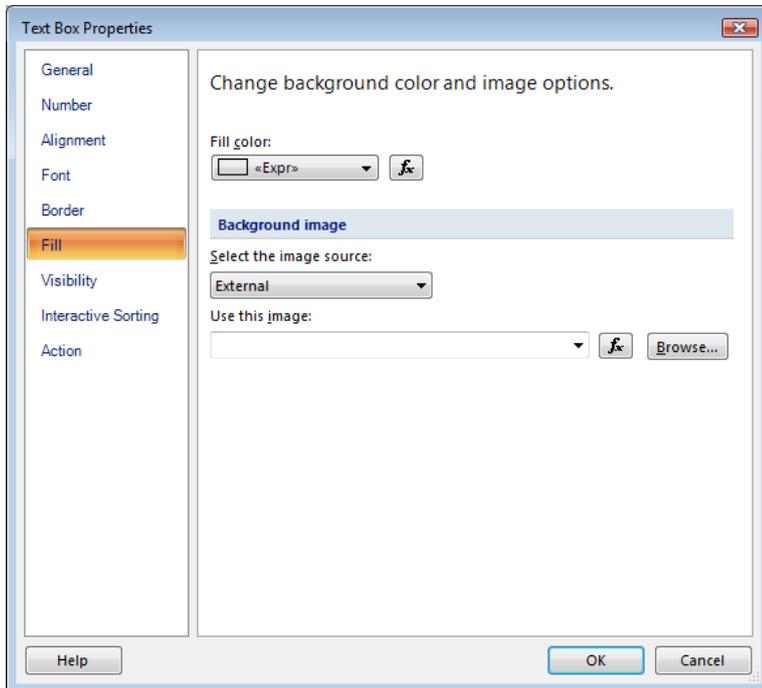
We have done it like this:



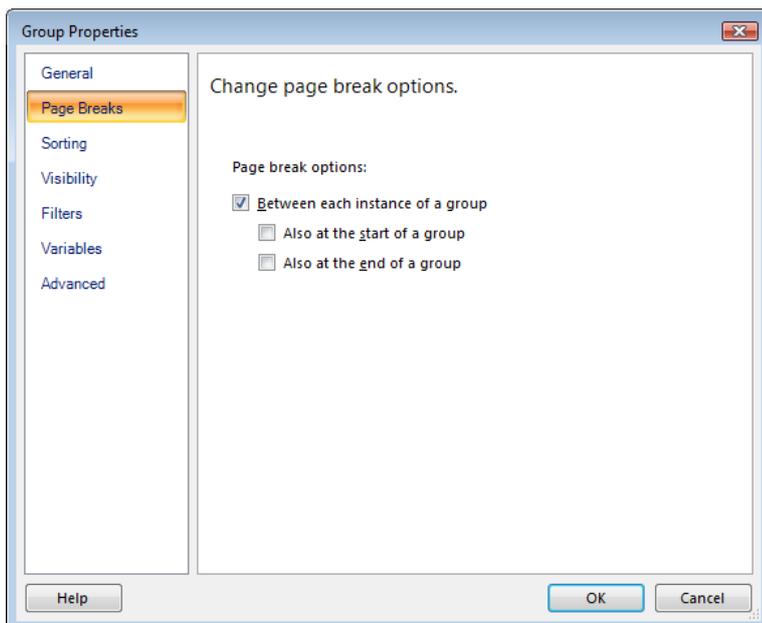
Now we will format the **Category** row with alternate colors, using this formula

```
=IIF(RunningValue(Fields!CategoryName.Value,COUNTDISTINCT,"Year") MOD 2 = 0,"PaleGreen","White")
```

- Right click the cell with [CategoryName].
- Click **Text Box Properties** ⇒ **Fill**.



- Click **fx** behind **Fill color**.
- Fill in the formula.
- Click **OK** twice.
- Run the report to check.
- Repeat the steps for all the cell of the CategoryName row.
- Create a page break after each group.



We should now see something like this:

Name				
Andrew Fuller				
1996				
Beverages	2226,4000	€2150,40	3,41%	16
Condiments	3721,6000	€3631,84	2,41%	5
Confections	2824,0000	€2820,00	0,14%	7
Dairy Products	2280,0000	€2112,00	7,37%	6
Grains/Cereals	1655,4000	€1539,90	6,98%	4
Meat/Poultry	7380,0000	€7040,40	4,60%	13
Produce	1346,8000	€1299,48	3,51%	6
Seafood	1400,5000	€1163,04	16,96%	4
	22834,7000	€21757,06	4,72%	8
1997				
Beverages	9279,5000	€9009,10	2,91%	9
Condiments	6967,5500	€6036,77	13,36%	9
Confections	11774,0500	€11706,13	0,58%	10
Dairy Products	13682,7000	€12440,05	9,08%	13
Grains/Cereals	7333,2000	€6838,80	6,74%	10
Meat/Poultry	12648,0500	€11978,59	5,29%	13
Produce	4780,0000	€4727,00	1,11%	9
Seafood	8493,5500	€7707,70	9,25%	10
	74958,6000	€70444,14	6,02%	11
1998				
Beverages	30523,5000	€29088,75	4,70%	(99474)
Condiments	5663,9500	€5182,06	8,51%	6

- Delete the top row.
- Merge the top cells except the first one.
- Put the cursor in upper left cell.
- Press **ENTER** twice.
- Type **Grand Total**.
- Right-click after **Grand Total**.
- Click create **Placeholder**.

Placeholder Properties

Change label, value, and markup options.

Label:

Value:

Tooltip:

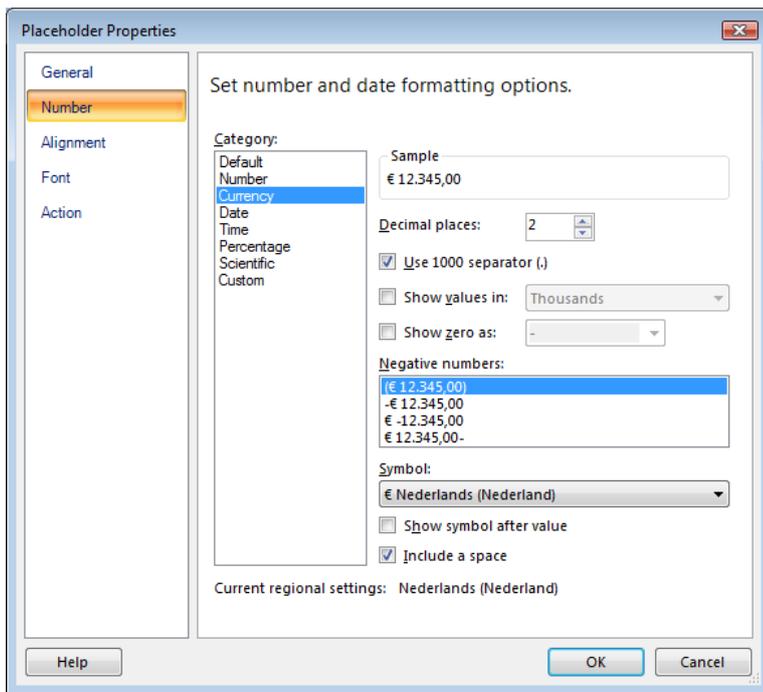
Markup type

None - Plain text only

HTML - Interpret HTML tags as styles

Help OK Cancel

- At **Value** fill in:
- [Sum(GrossSales)]**
- Click **Number**.
 - Fill it in like we see in the illustration.

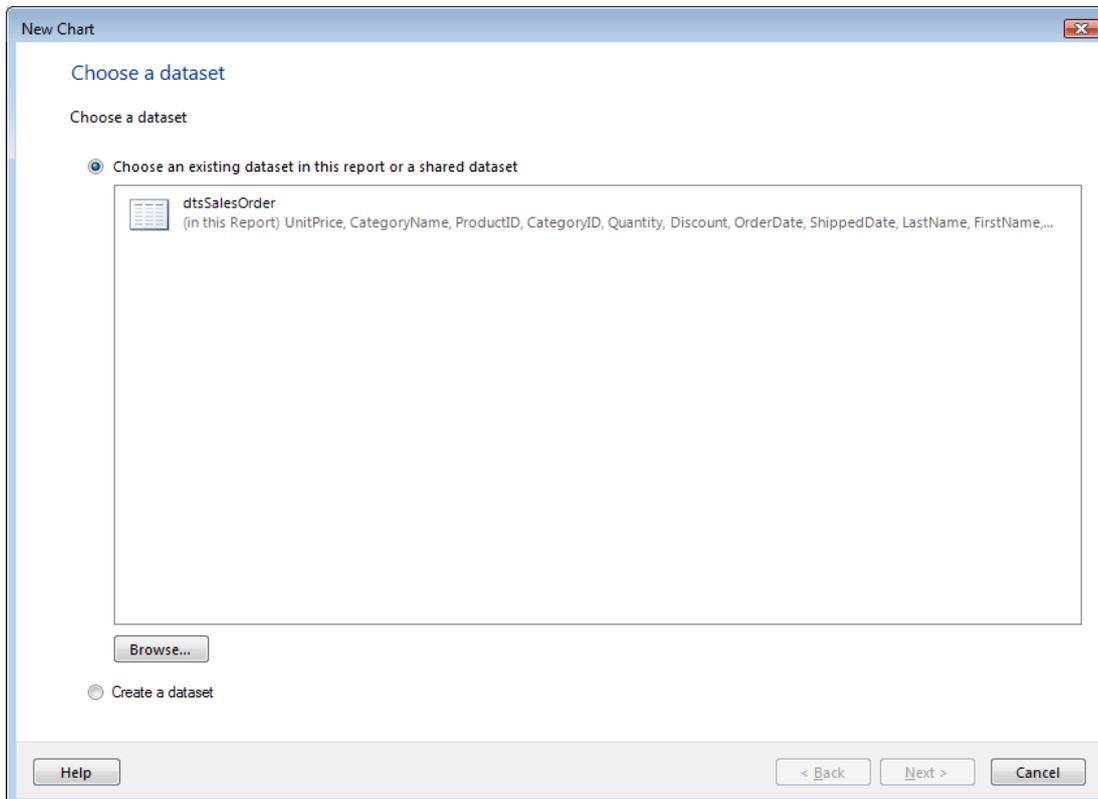


- Click **OK**.

9.2.3 Graph

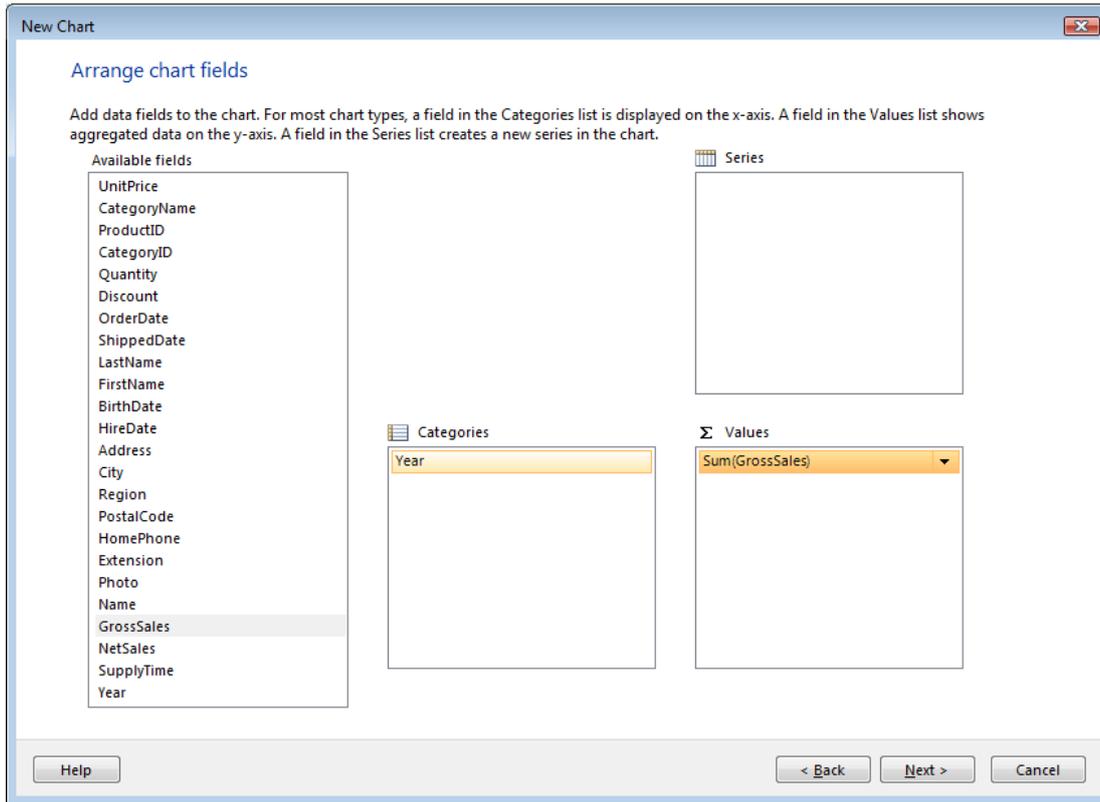
Now we will place a graph in the merged upper cells.

- Click the tab **Insert**.
- Click **Chart** ⇒ **Chart Wizard**.



- Click **dtsSalesOrder**.
- Click **Next**.
- Click **Column**.

We'll get:



- Fill it in like here.
- Click **Next**.
- Click **OK**.
- Drag the graph to the merged cells.
- Run the report.
- Save the report (**Report100.rdl**).

9.3 Linking scheme

