Evaluation Scheme:

|  |  |
| --- | --- |
| Customer Definition | 5 |
| Customer Data Set | 5 |
| Sales Definition | 5 |
| Sales Data Set | 5 |
| Customer Web Report +  Choice of charting mechanism | 5 |
| Sales Web Report  +  Choice of charting mechanism | 5 |
| Objectives | 10 |
| Conclusion | 10 |
| Overall Report | 10 |
| Total | 60 |

Notes for completing assignments:

In this assignment, you are going to develop a Customer Web based report and a Sales web based report. A substantial amount of IT resources goes into building reports. Reports are created for most of the cycles of the business process. There are daily, weekly, monthly, quarterly and end of year reports of all aspects of the business-like sales, customers, products, inventories.

A great deal of technological innovation has occurred to reduce the cost of building reports, to avoid the high cost of programming them from scratch.

In this assignment, you are going to build two web-based reports based on data residing in the database and create controls to visualize the data.

Your report should have at least the following sections:

Objectives:

In this section you research and document the advantages of creating reports as we did. What other reporting technologies exist?

Each “time for action” section:

Summarize the steps and provide the screen shots of the intermediate processes and summarize the results.

In the displaying the report data you can render the data using a number of different charting mechanisms. Please select two and justify the appropriateness of your choice.

Conclusion:

Your conclusion should reflect your experience performing the exercise: what was easy, difficult and obscure.

Developing a basic report

Designing and developing a basic report in SSRS is as easy as following the steps in a wizard. In this section, we will go through the scenario of creating our first SSRS report with the SSDT report wizard. You will learn about the structure of a Reporting Service project. Data sources will be used to connect to the source database, and then datasets will be explored which contain SQL queries to fetch data from the data source. We will go through the report layout to see how to configure the page and report layout.

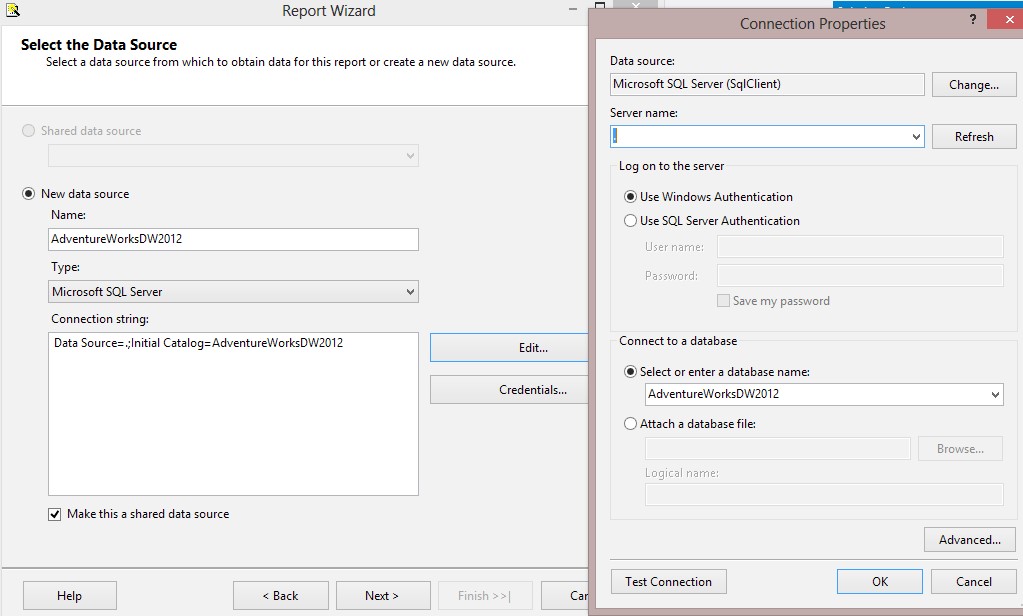
Time for action – creating our first report using SSRS

In this example, you will learn how to create a report server project with SSDT. We will use the AdventureWorks2012 database as source. Perform the following steps to create an SSRS report:

1. Open SSDT and create a new project. Under **Business Intelligence project templates**, choose **Report Server Project**. Name the project **Chapter 09 SSRS**.
2. The new project will be created with three folders: **Shared Data Sources**, **Shared Datasets**, and **Reports**.
3. Right-click on the **Reports** folder and click on the **Add New Report** option from the pop-up menu.
4. Now, the **Report Wizard** will appear. In the **Select Data Source** window, create a connection to the **AdventureWorksDW2012** database and name the connection **AdventureWorksDW2012**. Check the **Make this a Shared Data Source** checkbox option.

A shared data source can be used in multiple reports and datasets.

The following screenshot shows the **Report Wizard** and its connection properties:



1. In the **Design the Query** window, you can use the **Query Builder** option to build the query. For this example, use the following T-SQL code and paste it in the **Query String** textbox:

**SELECT DimProductCategory.EnglishProductCategoryName,**

**DimProductSubcategory.EnglishProductSubcategoryName,**

**DimProduct.EnglishProductName,**

**DimSalesTerritory.SalesTerritoryRegion,**

**DimSalesTerritory.SalesTerritoryCountry,**

**DimSalesTerritory.SalesTerritoryGroup,**

**FactInternetSales.SalesAmount**

**FROM DimProduct INNER JOIN**

**DimProductSubcategory ON**

**DimProduct.ProductSubcategoryKey =**

**DimProductSubcategory.ProductSubcategoryKey INNER JOIN**

**DimProductCategory ON**

**DimProductSubcategory.ProductCategoryKey =**

**DimProductCategory.ProductCategoryKey INNER JOIN**

**FactInternetSales ON**

**DimProduct.ProductKey =**

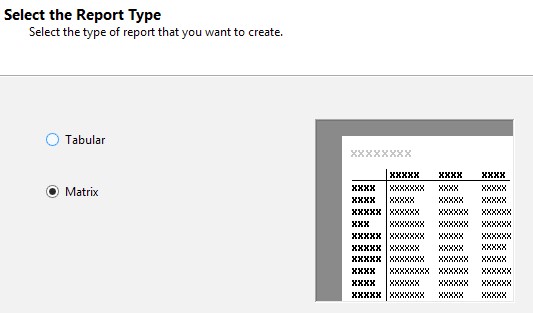
**FactInternetSales.ProductKey INNER JOIN**

**DimSalesTerritory ON**

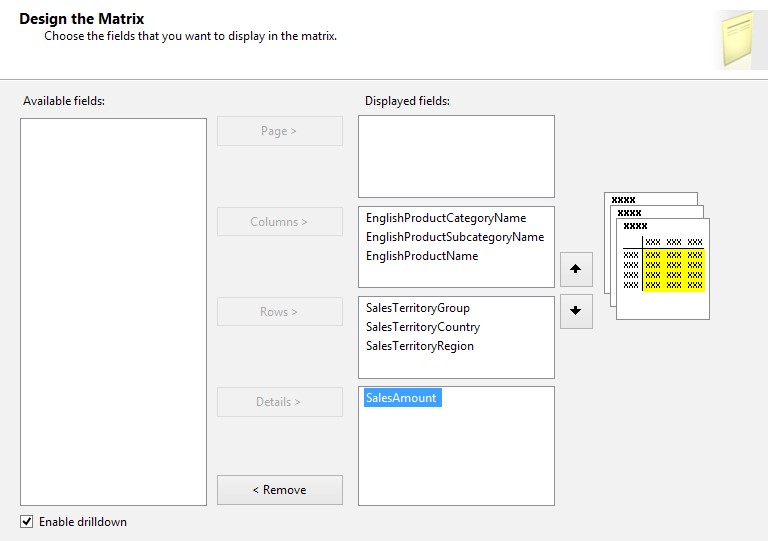
**FactInternetSales.SalesTerritoryKey =**

**DimSalesTerritory.SalesTerritoryKey**

1. In the **Select the Report Type** window, choose **Matrix**, as shown in the following screenshot:

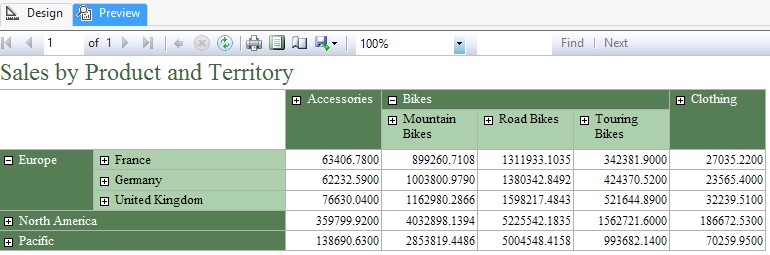


1. In the **Design the Matrix** window, add **EnglishProductCategoryName**, **EnglishProductSubcategoryName**, and **EnglishProductName** sequentially to the **Columns** section. You can add them by simply dragging-and-dropping them to the columns box or by clicking on the **Columns** button once for each item.
2. Add **SalesTerritoryGroup**, **SalesTerritoryCountry**, and **SalesTerritoryRegion** respectively to the **Rows** section.
3. Add **SalesAmount** to the **Details** section. Then, check the **Enable drilldown** option, as shown in the following screenshot:



1. In the **Choose the Matrix Style** window, choose the **Forest** option and continue.
2. In the last step of the wizard, rename the report to **Sales by Product and Territory**. Then, click on **Finish** to complete the wizard.
3. After finishing the wizard, you will see the report designer in SSDT with the generated report.
4. Click on the **Preview** tab and you will see the report result. The **Drilldown** option is available on the report, and you can drill down with expanding groups.

The following screenshot shows the **Preview** tab:

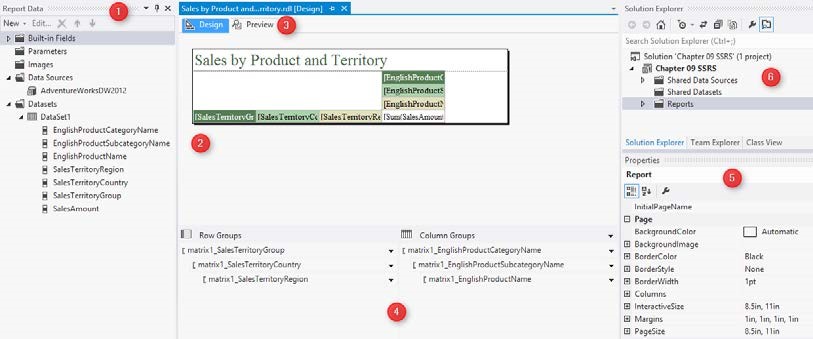


What just happened?

In this example, we used a simple **Add New Report** wizard to create a report using the SQL Server database. We went through the wizard steps, such as creating a data source connection; writing/building a SQL query for the dataset; and choosing a report style, layout, and positioning of the columns in the group.

The **Report Type** field can be chosen as **Tabular** or **Matrix**. The difference can be seen quite simply when you change these options in the small preview window beside it in the **Select the Report Type** window (step 6). The **Tabular** style shows headers only on columns, and data rows will be under it in the form of details. The **Matrix** style shows headers for both columns and rows. Choosing the right style depends on the type of grouping and items you want to show in the report.

After creating the report (step 12), SSDT displays the **Report Designer** menu. In SSDT, this is where report developers spend most of their time developing sophisticated and robust reports. In real-world scenarios, you are always required to develop the report much further than what the basic report wizard generates for you. But the report wizard is a good base for a report, from where you can then enhance it with the designer. The following screenshot shows the SSDT layout for Reporting Services projects:



The following are the explanations for each window in SSDT for Reporting Services projects:

* **Report Data**: This window shows information about data items in the report. Information such as data sources, datasets, query fields, parameters, and built-in fields is shown here.
* **Report**: This window will be used to change the layout of a report with items from **Toolbox** and data items from **Report Data**.
* **Preview**: This is the built-in **Preview** pane that displays a preview of the current report. The SSRS service doesn't need to be up and running to preview the current report.
* **Row Groups** and **Column Groups**: The attributes groups will be listed in this pane. You can reorder them here or change the configuration of the existing groups in this pane.
* **Properties**: This window shows the properties of the selected object in the report.
* **Solution Explorer**: This window shows the Reporting Services project solution structure.
* **Toolbox**: This window (not in the previous screenshot) contains report items such as textboxs, graphs, and KPIs that can be added to the report designer.

In this example, you learned how easy it is to generate a report with the drill-down functionality. You can also check the report export options in the **Preview** pane, which are exporting the report to PDF or printing the report. SSRS reports can be made printer friendly with very little effort.

Extended report development

The previous section showed you a very basic sample of reports. In this section, we will go through some customized development scenarios in reports, such as changing the sorting order of a column's data, changing the aggregation functions and parameter configurations, and adding charts.

Parameters

One of the most vital components of reports is parameterization. Parameters help end users filter reports and select the portion of data rows that they want. Also, applications that build on a report can interact with report parameters and send the parameters' values to the report from their GUI through code.

##### Time for action – adding parameters to a report

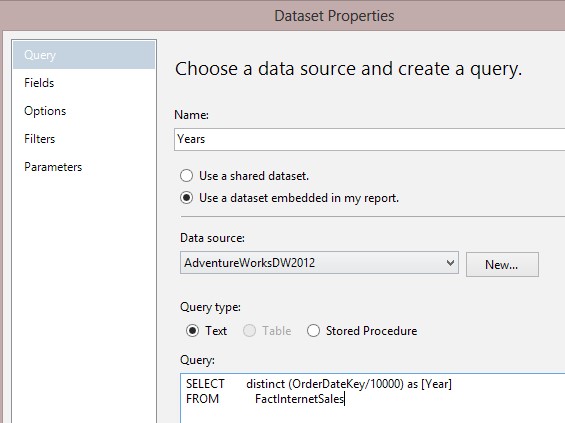
In this example, we will add the year and month parameters to the report generated in the previous section. You need to perform the following steps after the execution of the previous example to add parameters to a report:

1. Open the **Sales by Product and** **Territory** reports that were created in the previous example, and in the **Report Data** pane, right-click on **Dataset** and select **Add Dataset**.
2. In the **Dataset Properties** window, rename the dataset to **Years**.
3. Select the **Use dataset embedded in my report** option. Choose **Data source** as **AdventureWorksDW2012**. Then, in the query box, type in the following query to fetch all years from the **FactInternetSales** table:

**SELECT distinct (OrderDateKey/10000) as [Year]**

**FROM FactInternetSales**

The following screenshot shows the **Dataset Properties** window:



1. Close the **Dataset Properties** window and then right-click on **DataSet1** and select **Query**.
2. In the query designer window, add the following script line at the end of the **SELECT** query:

**WHERE (FactInternetSales.OrderDateKey / 10000 = @Year)**

1. After adding the preceding line of code, the final query looks like the following code snippet:

**SELECT DimProductCategory.EnglishProductCategoryName,**

**DimProductSubcategory.EnglishProductSubcategoryName,**

**DimProduct.EnglishProductName,**

**DimSalesTerritory.SalesTerritoryRegion,**

**DimSalesTerritory.SalesTerritoryCountry,**

**DimSalesTerritory.SalesTerritoryGroup,**

**FactInternetSales.SalesAmount**

**FROM DimProduct INNER JOIN**

**DimProductSubcategory ON**

**DimProduct.ProductSubcategoryKey =**

**DimProductSubcategory.ProductSubcategoryKey INNER JOIN**

**DimProductCategory ON**

**DimProductSubcategory.ProductCategoryKey =**

**DimProductCategory.ProductCategoryKey INNER JOIN**

**FactInternetSales ON DimProduct.ProductKey = FactInternetSales.**

**ProductKey INNER JOIN**

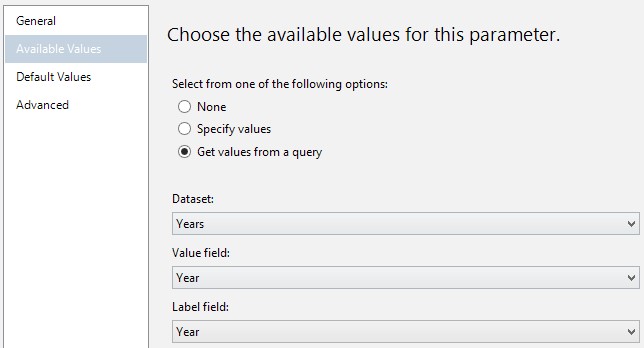
**DimSalesTerritory ON**

**FactInternetSales.SalesTerritoryKey =**

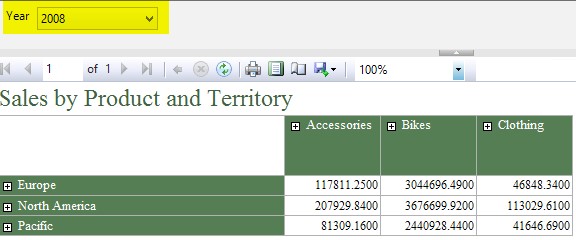
**DimSalesTerritory.SalesTerritoryKey**

**WHERE (FactInternetSales.OrderDateKey / 10000 = @Year)**

1. Close the **Query** window and go to the **Preview** window. You will see that the report asks for the value for **Year** in a textbox. Enter **2007** and click on **View Report**. Change the year and you will see that the report's data will change.
2. Now we want to change the user interface for the **Year** filter to be a drop-down list.
3. Go back to the **Report** designer, expand the **Parameters** folder in the **Report Data** pane, and double-click on the **Year** parameter.
4. Leave the **General** tab as is and go to **Available Values**. Choose the **Get values from a query** option, as shown in the following screenshot:



1. Choose **Dataset** as **Years** and select **Year** for both the **Value** and **Data** fields.
2. Close the **Parameter** window and go to the **Preview** pane. You will see that the textbox filter for **Year** changes to a drop-down list and you can see a list of years and can select any of them. The following screenshot shows the **Sales by Product and Territory** report for the year 2008:



1. Now we want to add the **Month** filter. The month will be selected after the year as a cascade filter.
2. Go back to the report designer and right-click on the **Datasets** folder in the **Report Data** pane and add a new dataset.
3. Rename the new dataset to **Months** and choose the embedded dataset option.

Select the data source and write the following query in the **Query** box:

**SELECT distinct**

**DATENAME(MONTH,convert(date,convert(varchar(8),OrderDateKey),**

**112)) as [MonthName]**

**,DATEPART(MONTH,convert(date,convert(varchar(8),OrderDateKey),**

**112)) as [MonthKey]**

**FROM FactInternetSales where OrderDateKey/10000=@Year**

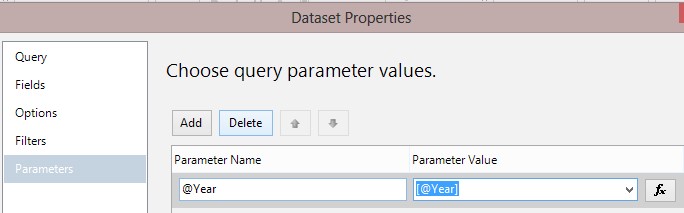
**order by DATEPART(MONTH,convert(date,convert**

**(varchar(8),OrderDateKey),**

**112))**

1. Go to the **Parameters** tab of the **Dataset Properties** window and choose the **[@**

**Year]** parameter in the **Parameter Value** box, as shown in the following screenshot:



1. Close the **Dataset Properties** window and edit the query for **DataSet1** by adding the following line in the query:

**AND ((FactInternetSales.OrderDateKey / 100 % 100) IN (@Month))**

1. After adding the preceding line of code, the following will be the final query of **DataSet1**:

**SELECT DimProductCategory.EnglishProductCategoryName,**

**DimProductSubcategory.EnglishProductSubcategoryName,**

**DimProduct.EnglishProductName,**

**DimSalesTerritory.SalesTerritoryRegion,**

**DimSalesTerritory.SalesTerritoryCountry,**

**DimSalesTerritory.SalesTerritoryGroup,**

**FactInternetSales.SalesAmount**

**FROM DimProduct INNER JOIN**

**DimProductSubcategory ON**

**DimProduct.ProductSubcategoryKey =**

**DimProductSubcategory.ProductSubcategoryKey INNER JOIN**

**DimProductCategory ON**

**DimProductSubcategory.ProductCategoryKey =**

**DimProductCategory.ProductCategoryKey INNER JOIN**

**FactInternetSales ON DimProduct.ProductKey = FactInternetSales.**

**ProductKey INNER JOIN**

**DimSalesTerritory ON**

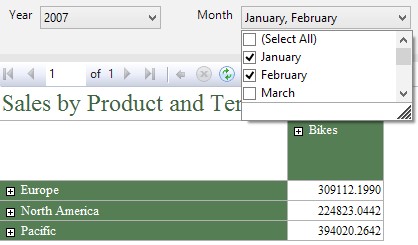
**FactInternetSales.SalesTerritoryKey =**

**DimSalesTerritory.SalesTerritoryKey**

**WHERE (FactInternetSales.OrderDateKey / 10000 = @Year)**

**AND ((FactInternetSales.OrderDateKey / 100 % 100) IN (@Month))**

1. Under the **Parameters** folder in the **Report Data** pane, double-click on the **Month** parameter to edit.
2. In the **General** tab, check the option **Allow multiple values**.
3. In the **Available Values** menu, choose **Get Values from a query**. Next, choose the **Months** dataset, select **MonthKey** as the **Value** field, and select **MonthName** as the **Label** field. ***22.*** Preview the report and select the year. You will now be able to choose the month. Note that you can choose multiple months, as shown in the following screenshot:



What just happened?

In this example, you learned how to add parameters to a report. Parameters will be added in the report GUI as an interface item and they will also be added to the dataset. Interface or report parameters can be customized in the **Parameter** properties window. You also learned how to cascade parameters in the report. Parameters can be customized to work as a single value, as multiple values, with a default value, and by selecting available values from a dataset.

Steps 1 to 3 show how to add a dataset manually to the report. The datasets created in these steps will be used in the following steps as the source for the available values of the **Year** parameter. The **Distinct** clause in the query will remove duplicate years.

In steps 4 to 6, we added the **Year** parameter to the dataset query of the main dataset. There are two types of parameters in SSRS reports: **Dataset** and **Report**. Dataset parameters will be added to datasets in a similar manner to how we've done so in these steps. To add a Dataset parameter, you only need to add the parameter to the dataset query. For this example, we added the parameter with an **@** prefix.

After adding the Dataset parameter, we are able to map the Dataset parameter to a Report parameter. If we don't map the Dataset parameter to a Report parameter, and if there is no Report parameter with the same name in that report, then a new Report parameter will be generated automatically.

Report parameters are interface options on the report that interact with the user. We can specify the data type of a Report parameter as either single value or multivalue. A Report parameter can read the available values from a dataset. As you've seen in step 7, the Report parameter's default interface is a textbox.

The available values of the Report parameter can be fetched from a dataset. You learned how to read the values of a dataset and feed it into the available values of the parameter in steps 9, 10, and 11. When a list of values comes from a dataset, then the Report parameter will be shown as a drop-down list and the user can choose the value from the list (step 12).

You also learned how to add a nested parameter in a report. We used the **Year** parameter as the main parameter, and then we've added the **Month** parameter after that. The **Month** filter, in this example, will be selectable only if the **Year** filter is set. So, we added the **Year** parameter as part of the **Months** dataset query (step 15). We've also picked two columns as the result of a query for this dataset: one for the month number and another for the month name. The **@Year** parameter in the dataset query of the **Months** dataset should be mapped to the **Year** parameter in the report; this mapping will be done in the **Parameters** tab in the **Dataset Properties** window (step 16).

The last part of this example was the configuration of the **Month** report parameter to be multivalue. The first part of the multivalue configuration of a parameter is to write the dataset query that uses this parameter in a way that it can handle multiple values. In step 18, you've seen that we've added the month criteria in a **Where** condition with the **IN** clause. We cannot use the equal operator for this criteria because equal can only be applied on a single value operator. We also changed the **Report** parameter properties to work as multiple value in steps 21 and 22.

Printing and page configuration

Reporting Services reports can be designed in a printer-friendly manner. There are properties in reports that are helpful in the configuration of the page size for printing. In this section, we will go through some page configuration options for the report.

Time for action – changing a page's properties

In this example, we will change the **PageSize** property of the report to reveal how the page size will affect the printing layout of the report. Perform the following steps:

1. Create a new report and use the same data source for it, **AdventureWorksDW2012**.

Then, use the following query as the query string:

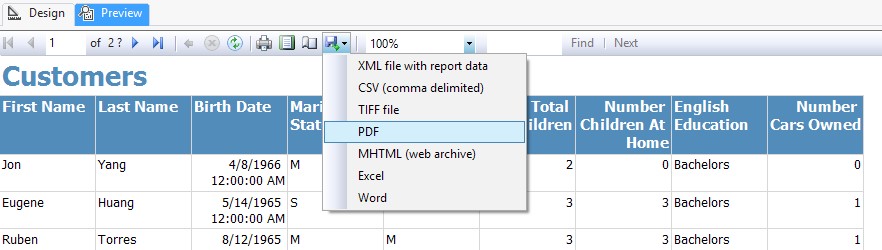
**SELECT FirstName, LastName, BirthDate, MaritalStatus,**

**Gender, TotalChildren, NumberChildrenAtHome,**

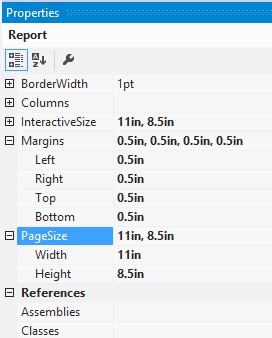
**EnglishEducation, NumberCarsOwned**

**FROM DimCustomer**

1. Choose **Tabular** as the report type. Next, select all columns to be in the **Details** section in the **Design the Table** window. Rename the report to **Customers** and complete the wizard.
2. Preview the report. In the **Preview** window, click on the **Export** button and export the report to PDF (as shown in the following screenshot):



1. Save the PDF file and then open it. You will notice that the number of pages is 1762 by default. Next, the pages exported to the portrait layout caused each report page to split into two PDF pages.
2. Go back to the report designer. In the **Properties** window, select the **Report** object. Then, expand the **PageSize** properties and change **Height** to **8.5in** and **Width** to **11in**. Also, change all margins to **0.5in**, as shown in the following screenshot:



1. Save the changes and preview the report. Export the report to PDF. This time, you will see that the number of pages is 1088. Each report page is exported in landscape in one PDF page.

What just happened?

In this example, you learned an easy way to perform page configuration in the report. We simply changed the page orientation and margin by changing a few properties. As a result, the report pages exported to PDF (or sent to a printer) will appear with those changes applied.

The **PageSize** properties define the layout of the physical page, which is a printer-friendly version of the report, and the properties will take action when the report is exported to PDF or sent to a printer. There is also another set of properties named **InteractiveSize**. These properties define the layout of the virtual page, which will be shown to users in the interactive mode.

Sorting and grouping

Reports show data rows, and one of the main requirements in reporting tools is the ability to order data rows and apply grouping and subtotals on groups. In this section, you will learn how to apply sorting and grouping in the report.

|  |
| --- |
| Time for action – applying ordering and grouping on the data |
|  |
| rows |

In this example, we will modify the customers' report generated in the previous example. We will add a **SalesAmount** column for each customer. We will apply sorting on the report and we will add a group for **NumberChildrenAtHome**. We will also add a subtotal for each group item. Perform the following steps:

1. In the **Customers** report, go to the **Report Data** pane and change the query of **DataSet1** to the following script:

**SELECT FirstName, LastName, BirthDate, MaritalStatus,**

**Gender, TotalChildren, NumberChildrenAtHome,**

**EnglishEducation, NumberCarsOwned**

**,sum(FactInternetSales.SalesAmount) as SalesAmount**

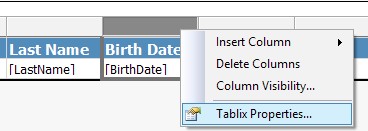
**FROM DimCustomer left outer join FactInternetSales**

**on DimCustomer.CustomerKey=FactInternetSales.CustomerKey group by FirstName, LastName, BirthDate,**

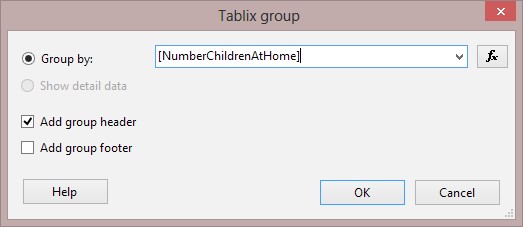
**MaritalStatus, Gender, TotalChildren, NumberChildrenAtHome,**

**EnglishEducation, NumberCarsOwned**

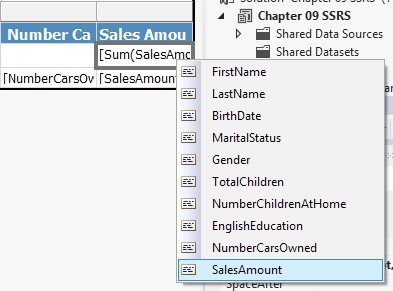
1. Now you will see the **SalesAmount** column under **DataSet1**. Drag-and-drop this column to the report designer exactly after the last column in the report.
2. Select the textbox under the **SalesAmount** column in the report designer. Then, in the **Properties** window, change the **Format** property to **#,#.##$**.
3. Select one of the column headers in the table in the report designer. Then, right-click on **Table Header Columns** and choose **Tablix Properties...** from the pop-up menu, as shown in the following screenshot:



1. In the **Tablix Properties…** window, go to the **Sorting** tab. Then, click on the **Add** button and choose the **SalesAmount** column with descending order (Z to A).
2. Preview the report. You will see that the **SalesAmount** column will be shown with a thousand separator, two decimal points, and a dollar sign. The report is ordered descending by sales amount.
3. Now we want to add interactive reporting to column headers.
4. Click on the **LastName** column header textbox (select the textbox itself – not the text in it) and then right-click on it and choose **Textbox Properties** from the pop-up menu.
5. In the **Textbox Properties** window, go to the **Interactive Sorting** tab, check **Enable interactive sorting on this text box**, and choose the **LastName** column in the **Sort by** section.
6. Preview the report. Now, you can click on the **LastName** column to change the sorting of rows (ascending or descending) based on this column.
7. Go back to the report designer. Click on the second row (the row after the column headers) and then right-click on the header of that row. Then, from the pop-up menu, select **Add Group**. Next, click on **Parent Group**.
8. In the **Tablix group** window, select the **[NumberChildrenAtHome]** column in the **Group by** drop-down list and check the **Add group header** option, as shown in the following screenshot:



1. You will see that a new row has been added to the **Tablix between header** and **Details** rows. This row is a group header row. Click on the textbox for the **SalesAmount** column in the group header row and then choose the **SalesAmount** expression from the pop-up menu, as shown in the following screenshot:



1. In the **Properties** window for this textbox, change the **Format** property to **#,#.##$**.
2. Preview the report. You will see the new group for **NumberOfChildrenAtHome**. The group header row shows the total sales amount for that group.
3. Now, we want to show the average number of cars for each group in the group header.
4. Right-click on the textbox for the **NumberCarsOwned** column in the group header row and select **Expression**.
5. In the **Expression** window, enter the following script:

**=Avg(Fields!NumberCarsOwned.Value)**

1. Close the **Expression** window. Change the **Format** property of this textbox to **#**.
2. Preview the report and you will see an average number of cars shown in the group header.

What just happened?

In this example, you learned how to enable interactive sorting for end users. You also saw how to apply a sort on data rows without user interaction. There is part of this example where grouping shows how to add a group on data rows as well as adding total aggregated measure as the group header. You also learned how to apply formatting on a textbox.

In the first two steps of this example, we modified a dataset query for the main report and we added a new data field to the report. Step 3 shows how to change the format string of a textbox to a numeral with a thousand separator and two decimal points. we changed the order of data rows in steps 4 and 5 with the **Sorting** tab of the **Tablix Properties** window. Steps 8 and 9 show how to enable interactive sorting on a column of the table in the report.

Steps 11 and 12 show how to add a group on the **NumberChildrenAtHome** data field for data rows. Groups can be added on either column or rows. There is also a **Grouping** pane in the report designer that shows the hierarchy of groups in rows and columns and provides an easier method to control groups on data fields. In this example, we've added the subtotal record for the group, and in step 13, we used that space to add the total of the sales amount for each group.

Steps 17 and 18 show how to add another textbox in the group header row with an expression. SSRS uses an expression language specific to itself. The expression language uses some built-in functions, and can be combined with data rows from the dataset and also some operators.

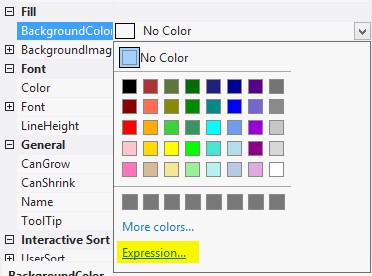
Expressions

SSRS expressions play a vital role in creating calculated fields in the report. Expressions are also an important part of dynamism in reports; for example, the background color of text can be changed based on an expression. In this section, we will have a very quick overview of expressions with an example that is based on the previous reporting project.

|  |
| --- |
| Time for action – changing the background color of data rows |
|  |
| based on expressions |

In this example, we want to distinguish records for male and female customers by changing the background colors of the row. For this purpose, we will write an expression and use that in the background color property of the data row using the following steps:

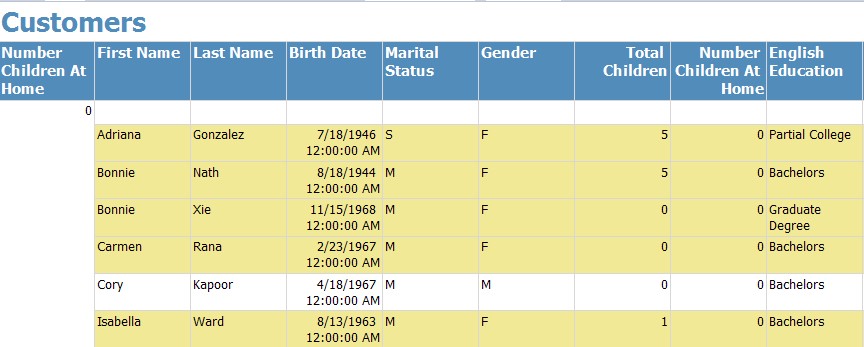
1. Go to the report designer of the **Customer** report from the previous example.
2. Click on the row header for the details record (third row in the Tablix).
3. In the **Properties** window, find the **Background Color** property and click on your desired color from the color picker menu.
4. Choose **Expressions** from the color picker pop up as shown in the following screenshot:



1. In the **Expression** editor window, enter the following script:

**=iif(Fields!Gender.Value="M","Light Green","Khaki")**

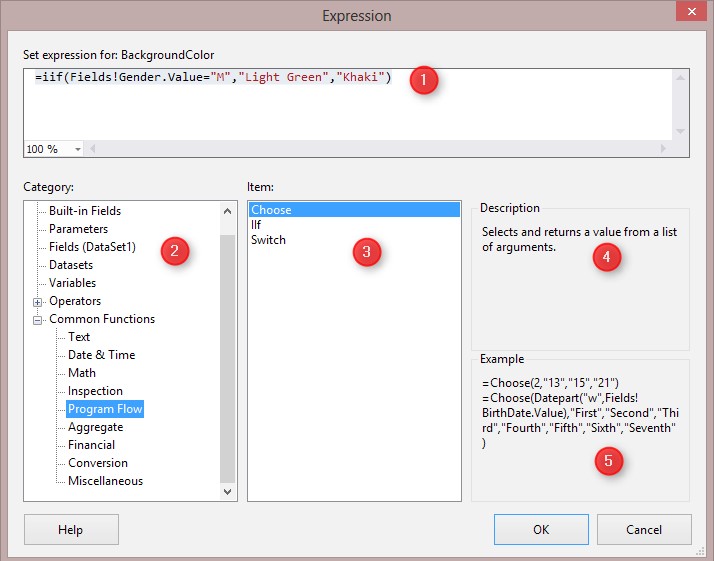
1. Close the **Expression** window and preview the report. You will see that male and female data rows are now different with regard to background color, as shown in the following screenshot:



What just happened?

This quick example explained how to change the background color of a set of textboxes (in a data row) with the result of an expression. As you can see in step 4, you can set the value of the background color property with an expression. Fortunately, most of the properties in SSRS report objects can be set with expressions. This feature empowers the dynamism of the report.

**Expression** is a functional language for writing scripts inside the report for collecting data with report items from dataset fields, or for creating calculated fields or scripts that can be used in the report item's properties (such as this example). Expression generates from a combination of built-in functions, operators, dataset fields, and/or variables. The following screenshot shows the **Expression** editor window and its sections:



The following is a list of the sections in the **Expression** editor in SSRS:

* Window for the script/text of the expression
* Category of functions / operators / variables
* Item list (the example shown in the screenshot is a list of functions under the **Program Flow** category)
* Description of the selected item
* Example scripts with the selected item

Explaining the full set of the **Expression** functions and operators is beyond the scope of this book. However, you can read any SSRS book to get more details about expressions. Online Microsoft MSDN books can also give you a lot of information on expressions, which can be read at **http://msdn.microsoft.com/en-us/library/dd220516.aspx**.

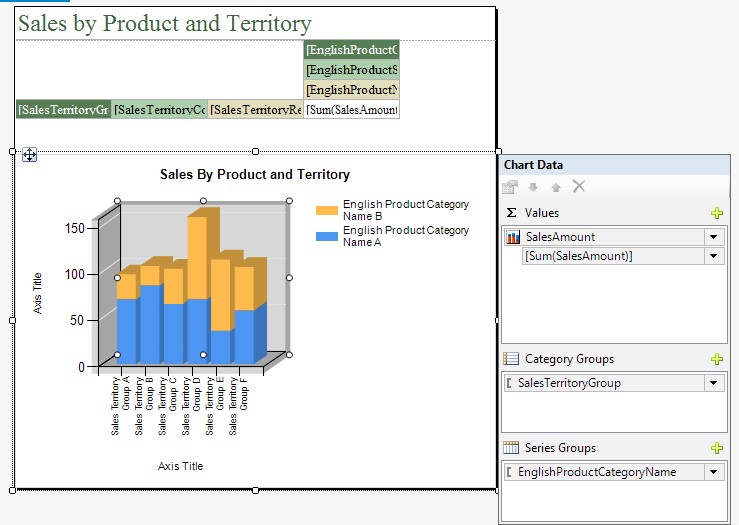
Adding charts

Charts, KPIs, and dashboards are some of the main components of analytical reporting. Reporting Services uses a wide range of charts and KPIs with highly configurable settings that provide a robust reporting platform for the end user.

##### Time for action – working with charts in Reporting Services

In this example, we will add a chart for the sales information in the **Sales by Product and Territory** report from the first example of this chapter. For simplicity, we will just create a stacked column chart with the default configuration from the existing dataset with the help of the following steps:

1. Open the report designer for the **Sales by Product and Territory** report.
2. Click on an empty area in the report designer and then go to the **Properties** window and choose a **Body** object.
3. Change the size of the **Body** object with these parameters: **Width** to **5in** and **Height** to **5in**.
4. Drag-and-drop a **Chart** object from **Toolbox** into the report designer under the **matrix** object.
5. Choose **Chart type** as **3-D Stacked Column**.
6. Change the title of the chart to **Sales by Product and Territory**.
7. Click on the chart shape and you will see the **Chart Data** pane appear on the right-hand side of the chart, as shown in the following screenshot:



1. From the **Report Data** pane, drag-and-drop **SalesAmount** into the **Values** section of **Chart Data**. Drag-and-drop **SalesTerritoryGroup** from **Report Data** into **Category Groups** of **Chart Data**. And finally, drag-and-drop **EnglishProductCategoryName** into **Series Groups**.
2. Preview the report for all months of the year 2008 and you will see that information in the chart right below the matrix.

What just happened?

This example showed how to work with charts in SSRS reports. In this example, we just used a very simple configuration by changing the **Chart Data** menu to show the functionality of charts in Reporting Services. Real-world charts would require more configurations on the chart properties; legend configurations; axis titles and settings; and color, size, and other configurations for the columns themselves.

As you've seen in the **Select Chart Type** dialog box (step 5), there are many types of charts supported by Reporting Services, such as column, bar, line, scatter, and so on. Every chart has a **Chart Data** pane such as that you configured in steps 7 and 8. **Chart Data** includes all data fields from the existing datasets that build the chart. There are many configurations that can be changed on the chart, legend, and/or axis with a right-click on each component of the chart.

Reporting Services also uses Gauges, Sparkline, and Data Bar, which are very useful as KPIs. It also uses maps to show geographical information. This chapter is not enough to explain all of these components, and I strongly recommend that you read BOL or Reporting Services books to increase your reporting skills with SSRS.

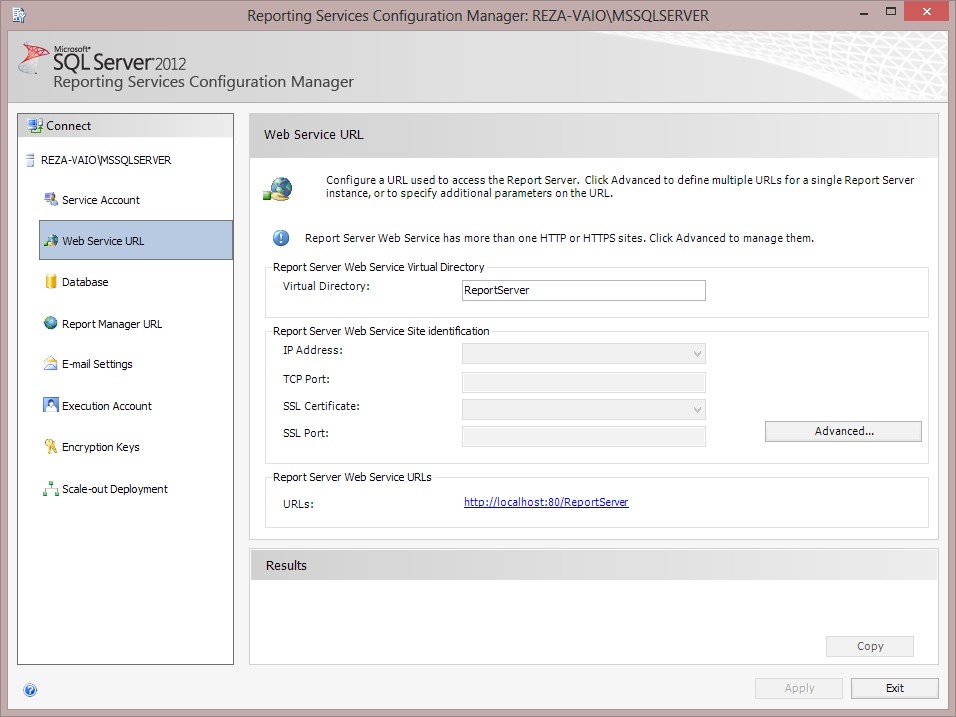
Deploying and configuring

Developed reports need to be deployed to UAT or production environments. SSRS reports can be deployed easily from SSDT. Reports can also be deployed from the WebUI to the **Report Manager**. There are two web applications for Reporting Services: **Report Server** for deployment and browsing reports and Report Manager for configuration, security, and administration tasks on reports and data sources. There is a tool for configuring SSRS components, named SQL Server Reporting Services Configuration Manager. In this section, we will deploy our reports into Report Server and then we will have a very quick look at **Report Manager** and Configuration Manager.

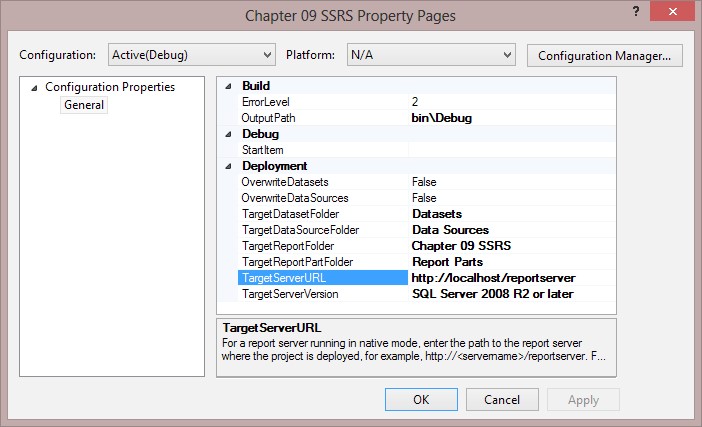
Time for action – deploying a report

In this example, we will first find the report server URL from Reporting Services Configuration Manager, and then we will use that URL to deploy reports from SSDT. Finally, we will browse those reports to see them from the Report Server web application by using the following steps:

1. Go to **Start** **|** **Microsoft SQL Server 2014** and expand **Configuration tools**. Then, click on **Reporting Services Configuration Manager**.
2. When **Reporting Services Configuration Manager** opens, it may ask for a connection. Just choose the default instance and connect.
3. You will see the service status that shows information about the SQL Server version, edition, and service status information.
4. Click on **Web Service URL**. The URL of the Report Server web application can be viewed or changed in this option, as shown in the following screenshot:



1. Go to **SSDT** and right-click on the Reporting Service project, and then from the pop-up menu, select **Properties**.
2. In the **Project Properties** window, change the **TargetServerURL** option to the URL of the report server that you picked from Reporting Services Configuration Manager. Then, close the **Properties** window.
3. Right-click on the project and click on **Deploy** to deploy the project, as shown in the following screenshot:



1. After successful deployment, go to the report server URL on Internet Explorer. In the report server web page, open the **Chapter 09 SSRS** folder and navigate to any of the reports there.