**SSRS Intermediate Assignment**

To date you have learned a great deal about creating and formatting reports, filtering and sub-totalling them, grouping values, and creating calculated fields. You can make grouped reports, charts and crosstab reports as well. You have learned a great deal of marketable skills and hopefully perspective. Most importantly the reports can all be dynamic, based on user-generated filtering (using parameters), and you have learned to move from one report to another by setting action spots.

This project analyzes sales representative performance. The DBA has placed the table with results into a shared database and has provided the following login information for WSU students:

|  |  |
| --- | --- |
| Database Server | cb-ot-devst03.ad.wsu.edu |
| Database | Featherman\_Analytics |
| User ID | mfstudent |
| Password | BIAnalyst |
| Table | SalesRepPerformancetoQuota |

Part 1 Scenario: Katie the new Sales Manager was recently hired in from an external consulting firm. Katie wants to perform a detailed analysis of salesrep performance to see if any trends are emerging and if any sales rep needs training or extra resources useful to close more sales. More than just a one-time salesrep analysis she wants a reporting system built that shows a dashboard of sales performance to quota complete with several levels of supporting analysis that can be drilled down into. Here are the specs.

Create a report, each will use a datasource connecting to the above mentioned database and each will need its own dataset created using the query builder to connect to the SalesRepPerformancetoQuota table shown above. Each individual report however will parameterize the query differently.  
  
**Report #1** should be filtered (with an @parameter) on year, so that Katie can select a year and see the quarterly sales performance for all sales reps (*the fieldname is employeeID or employeeKey*). So when the report user selects one year the results displayed are filtered by that year. So open the query of the dataset and add the parameter. In addition the data should be sorted (not parameterized or filtered) by EmployeeID (which can also be handled in the query builder or by implementing sorting in the tablix properties).   
  
You can just type in the available values for the parameter 2005, 2006, 2007, 2008. This report allows Katie to see all the sales rep (EmployeeID) performance for a specified year. Be sure to sort the table on sales rep, and then on quarter so that the quarterly performance is shown in correct order for each sales rep within the time period that was selected.   
  
Place the data onto the SSRS report using a table format and add an indicator for each row. Set the values for the indicator to <70.999 = red, <85.999 = yellow, and >=86 = green. When adding the indicator to the table, select expression and select the filed name rather than clicking in the saleskpi field. Check if the value is auto-set to COUNT(SalesKPI). If it does this behavior, then simply right-click and alter the expression of the indicator to ensure it reads =Fields!SalesKPI.Value. Also be sure the measurement units is set to numeric not percentage. Finally the top value can read =MAX(Fields!SalesKPI.Value). We will make a second report so this indicant can serve to drill down to the next level of detail.

Add a column chart to the top that shows the sales rep #’s along the horizontal axis (be sure to go into the horizontal axis properties and set the interval to 1) and shows the delta column as the values (for the categories use employee) – you can choose a different column for the values or if you have too much trouble a different chart all together. Add a nice title to the chart or report.  
  
After interacting with the data for 10 minutes, write your analysis of the sales rep performance for any particular year (after looking at each of the quarters for one year). Make a note of what additional supporting data you think the sales manager might need.

**Report #2** – In the same project, create a second report which will be a drill-down report. While the first report shows all employee (salesrep) sales for time periods, this report provides more detail to assess salesrep performance. The report will show the sales by category for a specific sales rep. report #1 passes in the salesrep (EmployeeID) with action spots on the top level report #1 on the employeeID, the sales kpi and the indicant. In each case you are drilling to this next report based on employeeID. Recall if the action is set on report elements they can pass a parameter value to the next report.   
  
This report will pull data from AdventureWorksDW2012, so in your project add a second datasource connecting to AdventureWorksDW2012 database. Next test this query in SSMS. If you test the query in SSRS you will have to remove the first line.

*If you do not understand any of the query below, it is fruitless to continue. Rather, please return to the T-SQL Primer document and investigate the lines that are not clear to you. If you do not have clarity of what this query means then you are shooting in the dark as you go forward…and just creating needless confusion.*

USE [AdventureWorksDW2012];

SELECT s.EmployeeKey, e.FirstName + ' ' + e.LastName AS Employee

,[SalesTerritoryCountry], [SalesTerritoryRegion] , pc.EnglishProductCategoryName

, DATEPART(year, s.OrderDate) as [Year]

, DATEPART(month, s.OrderDate) as [Month]

, SUM([OrderQuantity]) as [Total Sales]

FROM DimEmployee as e

INNER JOIN FactResellerSales as s ON e.EmployeeKey = s.[EmployeeKey]

INNER JOIN [dbo].[DimProduct] as p ON s.ProductKey = p.ProductKey

INNER JOIN [dbo].[DimProductSubcategory] as sc ON p.ProductSubcategoryKey = sc.ProductCategoryKey

INNER JOIN [dbo].[DimProductCategory] pc on sc.ProductCategoryKey = pc.ProductCategoryKey

INNER JOIN [dbo].[DimSalesTerritory] as t on t.[SalesTerritoryKey] = s.[SalesTerritoryKey]

WHERE s.EmployeeKey = 272

GROUP BY s.EmployeeKey, e.FirstName + ' ' + e.LastName

,[SalesTerritoryCountry], [SalesTerritoryRegion], pc.EnglishProductCategoryName

, DATEPART(year, s.OrderDate), DATEPART(month, s.OrderDate)

ORDER BY [SalesTerritoryCountry], [SalesTerritoryRegion], [Year], [Month]

You can see that the query is hard-coded for accepting sales for employee 272. Use this as your base query then parameterize the report (removing the hard-coded 272) with a new parameter for employeeID. Also remove the USE line from the query above when copying the SQL into the SSRS’ query builder.

*Add a matrix* control from the toolbox and place product category (down the rows), and year (across the columns) and show the unit values for the aggregated data for the selected employee. Add some charts.

*Add a second matrix control* and set of charts. Place one of the sales territory fields in the rows and category in the columns, and show unit totals as the aggregated data. You can experiment with adding the salesterritoryregion for the rows then adding a parent group on country or other groupings.

Now you are ready to parameterize the report and set the drill down from the first report. So add @EmployeeID parameters to the query, then go into the higher report and connect the drilldown (action feature) to pass in the value. Add the action from 1) the indicant, 2) and KPI field and 3) the employee name.  
  
Higher credit given for adding a heading using a textbox that is dynamically populated using the parameter values such as “Sales Performance for Joe Cool”

Note: While this is helpful information it can be improved with providing some additional information such as sales for the prior quarter or the same quarter for the prior year. The LAG function and PARTITION BY windowing functions will be demonstrated at a later date to provide this functionality. Another suggestion is to use the PIVOT() TSQL construct to create the matrix (crosstab) data format in the data, rather than the list that the GROUP BY function outputs. PIVOT() will also be demonstrated at a later date.  
--------------------------------------- end of part 1, beginning of part 2 ---------------------------------------------------  
  
R**eport** #3 that uses the same query as the datasource but will be parameterized on salerep (empID) and not year. The report should show all the performance for one salesrep and therefore should be filtered (@parameterized) on empID. So open the query of the new dataset in your new report and add the parameter. (Sort –not parameterize- the data by time period – you can accomplish this in the query builder or the by changing the tablix properties).

This report will need a second data set of employee names and IDs’s. This information is not in Featherman\_Analytics database. You will need to use a datasource that points to AdventureWorksDW2012 to get the values. In this report you will need a second dataset filled with the empID and name from the dimEmployees table (the name is the labelvalue). You can add a concatenated version of the name using the provided query below. There are hundreds of employees in the dimEmployees table but there are only 17 sales reps. So we add the inner join to display to filter down the results only to the employee id’s that have made a sale recorded in the factresellersales table (aka the salesreps). Finally we add the filter DISTINCT to reduce down the list of sales reps.

SELECT DISTINCT DimEmployee.EmployeeKey, DimEmployee.FirstName + ' ' +   
DimEmployee.LastName AS Employee

FROM DimEmployee

INNER JOIN FactResellerSales ON DimEmployee.EmployeeKey = FactResellerSales.EmployeeKey

Again make a nice parameterized report that shows the sales performance for any selected sales rep (*EmployeeID or EmployeeKey*) over time. Sort the report on time period. Use the same indicator value settings as before. This report will drill down to new report #4 – which is a more detailed report based on indicator value.  
  
Add a line chart to show the KPI values over time (be sure the values are in correct sorted order). Again be sure to go into the horizontal axis properties and set the interval to 1. After interacting with the data for 10 minutes, write your analysis of the sales rep performance for a couple of sales reps (after looking at each of the quarters for one year). Make a note of what additional supporting data you think the sales manager might need when making an evaluation of the salesrep performance.  
  
Higher credit given if you make the report multi-select so that the table and chart can be used to compare the performance of two sales reps. Be careful however because if you use multi-select, the parameter must be created in the report and implemented in the table and chart; rather than in the query. For a multi-select parameter you do not add the parameter to the query.

Report #4 – Add another drill-down report, which drills down from report #3. This report is completely of your own design, however connect it to the report #2. Examine the second report you made which shows all the sales performance for a selected employee and envision where to add action sorts. Add a report with more detail to assist the new marketing manager.