

Imports System.Data

Imports System.Data.SqlClient

Public Class Charting1

Inherits System.Web.UI.Page

'This project demonstrates 1)the usefulness of the GroupBy SQL Statement, 2) the use of simple column and line charts in Visual Studio. The free PowerBI account we have access to only refreshes the data and therefore charts once per day. The $10/month version of PowerBI provides live data updates

'This document does not intend to explain the GROUP BY statement, other modules discuss that analytics tool. If you already know how to build datasets with that function, then you can see that the columns of compiled data that the GROUP BY create are displayed as columns or lines on a chart.

Public Shared Con As New SqlConnection("Data Source=cb-ot-devst03.ad.wsu.edu;Initial Catalog=Featherman\_Analytics;Persist Security Info=True; User ID=mfstudent; Password=BIanalyst")

#Region "Column Chart"

Protected Sub DropDownList1\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles DropDownList1.SelectedIndexChanged

'Please check the SQL Glossary and web, this is how you do a GROUP BY SQL Statement. Its a pretty darn good trick to be able to do this and show the results in a gridview so easily. Hopefully you will try to group on differenct columns, (here Region), another time by month (extracted from the Date field).

Dim GroupByRegionDA As New SqlDataAdapter("SELECT Region, FORMAT(SUM(Total\_Sale), 'N0') AS [RegionalTotal], COUNT([Sale\_ID]) AS [# Sales]

,FORMAT(SUM(Total\_Sale)/COUNT([Sale\_ID]), 'N0') AS [Avg. $ale] FROM featherman.sales WHERE Year = @p1 GROUP BY Region ORDER BY Region", Con)

Dim RegionTable As New DataTable

'the achilles heel of the procedure is that the SQL query MUST have a value for the year passed in from the web page. There is no default value specified (an interesting idea) so we need to stop the program.

If DropDownList1.SelectedIndex = -1 Then

Response.Write("Select a Year")

Exit Sub

End If

'clear out any old data from the in-memory data structure as necessary

If RegionTable.Rows.Count > 0 Then

RegionTable.Rows.Clear()

End If

'set up the parameter which takes the user selected year and puts that value into the SQL statement as the critia for the where statement

With GroupByRegionDA.SelectCommand.Parameters

.Clear()

.AddWithValue("@p1", Convert.ToInt16(DropDownList1.SelectedItem.Text))

End With

'now run the SQL statement over the connection to the database, pull the data into the table, and display it in the gridview.

Try

GroupByRegionDA.Fill(RegionTable)

With GridView1

.DataSource = RegionTable

.DataBind()

End With

'the charts also need their datasource identified and binding needed

With Chart1

.DataSource = RegionTable

.DataBind()

End With

With Chart2

.DataSource = RegionTable

.DataBind()

End With

Catch ex As Exception

Response.Write(ex.Message)

End Try

'---------------------------------- this section builds the chart. Just modify the fields as needed -------------------------------------------------

'You can specify the titles by passing in a variable. X axis is across the bottom, Y axis is vertical

Chart1.Titles.Add("Total Revenue by Region for " & DropDownList1.SelectedItem.Text)

With Chart1.ChartAreas(0)

.AxisX.Title = "Regions"

.AxisY.Title = "$ales for Region"

.Area3DStyle.Enable3D = True 'this is optional, but 3D charts are a little more interesting

End With

'each chart has series of data, usually one which is created for you. but you can add more than one (such as the third procedure that produces a line chart with 4 lines)

With Chart1.Series("Series1")

'make it a column chart - there are many different kinds.

'now we tell the chart that it should display data from the retrieved datatable that was created in the dataAdapter.fill command above (which creates a table with condensed database data). Then we specify the X axis values which are the categories (here the region) and we specify the Y axis values (here the totals). It is subtle but important to know that here the data in the regionaltotal column must already be compiled and totalled. The chart is not doing any compiling or summing of the data. The GROUP BY () query is doing the aggregation work, the chart control is only displaying the compiled numbers that are in the gridview.

'Finally we .databind the chart - similar to how we did this for the gridview (this tells the control to display the data. The X and y data values come from your SQL Query. There are many chart types (change the last property of the next line to see them).

.ChartType = DataVisualization.Charting.SeriesChartType.Column

.XValueMember = "Region" 'verify that this is a field name from the SQL query

.YValueMembers = "RegionalTotal" 'verify that this is a field name from the SQL query

End With

'------------------the first chart shows $ revenue. More important is profit after cost of business operations, but we do not have that data. So lets look at average invoice amount. This next code is redundant. there is nothin new. the point is that the analyst or manager should see many charts to understand a problem, not just one.

Chart2.Titles.Add("Average $ale Amount for Each Region for " & DropDownList1.SelectedItem.Text)

With Chart2.ChartAreas(0)

.AxisX.Title = "Regions"

.AxisY.Title = "Average $ales for Region"

.Area3DStyle.Enable3D = True

End With

With Chart2.Series("Series1")

.ChartType = DataVisualization.Charting.SeriesChartType.Column

.XValueMember = "Region" 'verify that this is a field name from the SQL query

.YValueMembers = "Avg. $ale" 'verify that this is a field name from the SQL query

End With

End Sub

#End Region

Protected Sub DropDownList2\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles DropDownList2.SelectedIndexChanged

'this group by SQL statement sums up sales for the one selected region for each of the years in the dataset. The data is compiled and put into buckets for each year (other time periods ar also possible). The data is then displayed in a line chart. This is basically the same chart making code however rather a cross-sectional look at the data (one year) its over time as above without some formatting and a fancy title.

Dim GroupByDA As New SqlDataAdapter("SELECT Year, FORMAT(SUM(Total\_Sale), 'N0') AS [Sales Totals], COUNT([Sale\_ID]) AS [# Sales]

,FORMAT(SUM(Total\_Sale)/COUNT([Sale\_ID]), 'N0') AS [Avg. $ale] FROM featherman.sales WHERE Region = @p1 GROUP BY Year ORDER BY Year", Con)

Dim YearData As New DataTable

'we need to make sure a region is selected so we use error checking for that

If DropDownList2.SelectedIndex = -1 Then

Response.Write("Select a Region")

Exit Sub

End If

'this is how we get the user selected value from the dropdownlist an pass it to the red SQL statement that the data adapter will run.

With GroupByDA.SelectCommand.Parameters

.Clear()

.AddWithValue("@p1", DropDownList2.SelectedValue)

End With

Try 'retrieve and compile and compile the data into the data table. Next tell the charts and grdview control where to get their data from.

GroupByDA.Fill(YearData)

With GridView1

.DataSource = YearData

.DataBind()

End With

With Chart1

.DataSource = YearData

.DataBind()

End With

With Chart2

.DataSource = YearData

.DataBind()

End With

Catch ex As Exception

Response.Write(ex.Message)

End Try

Chart1.Titles.Add("Sales Summary for " & DropDownList2.SelectedItem.Text & " Region")

'see now that the titles are passed in and dynamically set?

With Chart1.ChartAreas(0)

.AxisX.Title = DropDownList2.SelectedItem.Text

.AxisY.Title = "# Sales for " & DropDownList2.SelectedItem.Text

End With

'now we tell the chart that it should display data from the retrieved datatable that was created in the dataAdapter.fill command above (which creates a table which is a copy of the database data). Then we specify the X axis values which are the categories across the X axis (here the year) and we specify the Y axis values (here the totals). Finally we .databind the chart - similar to how we did this for the gridview (this tells the control to display the data. The X and y data values come from your SQL Query

Chart1.Series("Series1").XValueMember = "Year"

Chart1.Series("Series1").YValueMembers = "Sales Totals"

Chart1.Series("Series1").ChartType = DataVisualization.Charting.SeriesChartType.Line

'----------second chart-------------------------------

Chart2.Titles.Add("Number Sales per year " & DropDownList2.SelectedItem.Text & " Region")

'see now that the titles are passed in and dynamically set?

With Chart2.ChartAreas(0)

.AxisX.Title = DropDownList2.SelectedItem.Text

.AxisY.Title = "# Sales for " & DropDownList2.SelectedItem.Text

End With

'now we tell the chart that it should display data from the retrieved datatable that was created in the dataAdapter.fill command above (which creates a table which is a copy of the database data). Then we specify the X axis values which are the categories across the X axis (here the year) and we specify the Y axis values (here the totals). Finally we .databind the chart - similar to how we did this for the gridview (this tells the control to display the data. The X and y data values come from your SQL Query

With Chart2.Series("Series1")

.XValueMember = "Year"

.YValueMembers = "# Sales" 'notice that you do not have to use brackets for column names that have spaces.

.ChartType = DataVisualization.Charting.SeriesChartType.Line

End With

End Sub

Protected Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Dim PivotDA As New SqlDataAdapter("SELECT \* FROM (SELECT [Region], [Year], [Total\_Sale] FROM [featherman].[sales]

) AS BaseDataTable PIVOT (SUM([Total\_Sale]) FOR Region IN (North, South, East, West) ) AS PivotTable ORDER BY [YEAR]", Con)

Dim PivotTable As New DataTable

'clear out any old data as necessary

If PivotTable.Rows.Count > 0 Then

PivotTable.Rows.Clear()

End If

'now run the SQL statement over the connection to the database, pull the data into the table, and display it in the gridview.

Try

PivotDA.Fill(PivotTable)

With GridView1

.DataSource = PivotTable

.DataBind()

End With

'the chart also has a datasource and binding needed

With Chart1

.Titles.Add("Sales Summary by Region and Year")

.DataSource = PivotTable

.DataBind()

End With

Catch ex As Exception

Response.Write(ex.Message)

End Try

'---------------------------------- this section builds the chart. Just modify the fields as needed -------------------------------------------------

'You can specify the titles by passing in a variable. X axis is across the bottom, Y axis is vertical

With Chart1.ChartAreas(0)

.AxisX.Title = "Year"

.AxisY.Title = "$ales for Region"

.Area3DStyle.Enable3D = True

End With

'each chart has series of data, usually one, but you can add more than one (such as in a line chart with 4 lines)

'there are other ways to code this but the pivot query will make the 4 columns of data, and this code controls whether or not one of the columns is displayed (verify in the girdview that each column is for one of the regions).

If CheckBoxList1.Items(0).Selected = True Then

With Chart1.Series("Series1")

.ChartType = DataVisualization.Charting.SeriesChartType.Line

.XValueMember = "Year" 'verify that this is a field name from the SQL query

.YValueMembers = "North" 'verify that this is a field name from the SQL query

End With

End If

If CheckBoxList1.Items(1).Selected = True Then

Chart1.Series.Add("Series2")

With Chart1.Series("Series2")

.ChartType = DataVisualization.Charting.SeriesChartType.Line

.XValueMember = "Year" 'verify that this is a field name from the SQL query

.YValueMembers = "South" 'verify that this is a field name from the SQL query

End With

End If

If CheckBoxList1.Items(2).Selected = True Then

Chart1.Series.Add("Series3")

With Chart1.Series("Series3")

.ChartType = DataVisualization.Charting.SeriesChartType.Line

.XValueMember = "Year" 'verify that this is a field name from the SQL query

.YValueMembers = "East" 'verify that this is a field name from the SQL query

End With

End If

If CheckBoxList1.Items(3).Selected = True Then

Chart1.Series.Add("Series4")

With Chart1.Series("Series4")

.ChartType = DataVisualization.Charting.SeriesChartType.Line

.XValueMember = "Year" 'verify that this is a field name from the SQL query

.YValueMembers = "West" 'verify that this is a field name from the SQL query

End With

End If

End Sub

#Region "Load a DDL at program start-up"

Private Sub Charting1\_Init(sender As Object, e As EventArgs) Handles Me.Init

'this query pulls the years into a dropdown list.

Dim GetYearsDA As New SqlDataAdapter("SELECT DISTINCT(YEAR) as [Year] FROM featherman.sales ORDER BY [Year]", Con)

Dim YearsTable As New DataTable

Try ' the data adapter fetches the data by running the red query code above over the database connection specified.

GetYearsDA.Fill(YearsTable)

With DropDownList1 'this is how we

.DataSource = YearsTable

.DataTextField = "Year"

.DataBind()

End With

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

#End Region

End Class