Distilling More Analytics From Transaction DataTables

<http://cb-ot-devst05.ad.wsu.edu/featherman/NewFor2018/IntermediateDataTables2.aspx>

This version of the donations program adds three important enhancements. While they seem simple the functionality and coding techniques are a leap forward in sophistication and cool.

1. First the table.compute function is used with a filtering criteria on department, which is a little tricky but very helpful. We are able to pass the major from the radiobutton list into the table.compute filtering syntax.
2. The second major upgrade is that analytics are performed by compiling and distilling data from the detail transactions table and adding the summary analytics as rows into a new summary level datatable at a different level of granularity. Whereas the donations transaction data are at the highest level of granularity (detail), the summary analytics data is at a lower level of detail (higher level of summarization). While there could be hundreds of donations for each of the major, the transaction data is compiled for each major to just one row. This functionality is similar to SQL’s GROUP BY query which is covered in another module.

Each column within the new row is the result of calculations, so the entire process of analytics is a condensing operation. Table.compute functions are used to compile counts, sums and averages of a column of data. Here averages are calculated in a different manner and displayed in the gridview control.
3. The third major upgrade of this version is the use of dataviews to filter the transaction rows shown in the transactions table. The rows are already in RAM of the webpage so the dataview is a quick way to filter the transactions that are displayed in the gridview. You can for example see all the transactions for the Accounting major.

The analytics performed here should give you lots of ideas for your own project development. In particular the demonstration that for one row you can pull data from a set of columns to perform additional analytics is very interesting and huge. You the program designer can then decide what data needs to be put into columns of a row (we used table.compute) and that those summary metrics can be used in further analytics is a conceptual breakthrough.

 Use the data management and data analytics features here to work on your final project.

Imports System.Data

'This program adds three very important enhancements over the prior version. First the table.compute function is used with a filtering criteria on department, that is a little tricky but very helpful. The second major upgrade is that analytics are performed by adding rows to a new datatable at a different level of granularity. Whereas the transactions are stored in a details table, this project calculates compiled business metrics at a lower level of granularity, the major level (ie accounting, management etc.). When analytics are performed one new row is added to the summary table for each major.

'Each column within the new row is the result of calculations, so the entire process of analytics is a condensing operation. Table.compute functions are used to compile counts and sums of transactions. Then averages and comparison to goals are calculated and displayed in the gridview control.

'The third major upgrade of this version is the ease of filtering the transaction rows shown in the transactions table. A procedure is shown that uses a dataview to filter the transactions datatable. The rows are already in RAM of the webpage so the dataview is a quick way to filter the transactions that are displayed in the gridview. You can for example see all the transactions for the Accounting major.

'The analytics performed here should give you lots of ideas for your own project development. Use the features here to work on your final project.

Public Class IntermediateDataTables2

 Inherits System.Web.UI.Page

 'These are page level variables that are used to accumulate totals for each category

 Public Shared TransactionsTable, AnalyticsDataTable As New DataTable

 'these next variables are used to keep running totals for the entire set of donations in the current session of using the web page. These totals and average are updated every time a new transaction row is entered. In the analytics section the totals for each department are compared to these totals for the entire set of transactions. These values keep incrementing while the web page is open, but get reset when the web page is closed and started a new. A future application is tied to a SQL Server database to resolve this limitation.

 Public Shared gdecTotal As Decimal = 0

 Public Shared gdecAverage As Decimal = 0

#Region "Log in"

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

 'here we create the datatables for data storage. This code uses a nested if, so the code only works if the password was entered first.

 If txtPassword.Text = "swordfish" Then

 Panel1.Visible = True

 Panel2.Visible = True

 Else : Response.Write("go fish")

 End If

 End Sub

#End Region

#Region "Create Datatables"

 Private Sub IntermediateDataTables2\_Init(sender As Object, e As EventArgs) Handles Me.Init

 'Just in case the columns of either table were already created then exit, do not create them again

 If TransactionsTable.Columns.Count > 0 Then Exit Sub

 If AnalyticsDataTable.Columns.Count > 0 Then Exit Sub

 With TransactionsTable.Columns

 .Add("TransactionID", Type.GetType("System.Int32"))

 .Add("CustomerName", Type.GetType("System.String"))

 .Add("Major", Type.GetType("System.String"))

 .Add("Donation", Type.GetType("System.Decimal"))

 .Add("TADate", Type.GetType("System.String"))

 End With

 'Again the first column will automagically set its value. Note these values restart at 1 each time the webpage opens in a new browser, so a different approach would be used if saving data directly to a SQL Server database.

 With TransactionsTable.Columns("TransactionID")

 .AutoIncrement = True

 .AutoIncrementSeed = 1

 .AutoIncrementStep = 1

 End With

 'Interestingly in another procedure, when performing calculations the rows are added to the AnalyticsDataTable. The first column from the list below is hard-coded written in (accounting, management, MIS), the 2nd and 3rd are the result of a calculation, and the rest are calculated (averages and comparison to averages) only if there are donations in the NumberDonations column, to prevent a divide by 0 error. So the columns for this summary analytics table are created here and the rows are added with the departmental analytics in another procedure.

 With AnalyticsDataTable.Columns

 .Add("Major", Type.GetType("System.String"))

 .Add("NumberDonations", Type.GetType("System.Int32"))

 .Add("TotalDonations", Type.GetType("System.Decimal"))

 .Add("%Total", Type.GetType("System.String"))

 .Add("AvgDonation", Type.GetType("System.Decimal"))

 .Add("CompareToAvg", Type.GetType("System.String"))

 End With

 'This next line is the else, end if from the first if statement. If the password was typed in wrong you get the message here to

 End Sub

#End Region

#Region "Data Entry"

 Protected Sub Button4\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button4.Click

 'here we add rows of transaction data (individual donations) to a transactions table. You are learning the fundamentals of data management in a database or data warehouse. You have a transactions table (also known as the fact table) where all the detail transactions go for one process. You would use the same approach when saving data directly to SQL Server database, put detail transactions in one table, put descriptive information in other tables called dimension tables (not covered yet). So just know you are building up your knowledge of data models and data management.

 'again we create a new row at the bottom of the transactions table and one by one fill each of the columns (named in red) for that one new row. Finally we will add that new row to the table and display the data structure in the gridview control so you acan see all the rows including the new row.

 Dim dr As DataRow = TransactionsTable.NewRow

 'check for data entry errors, if any of the data entry controls are not filled in with suitable data then stop the program.

 If txtCustomer.Text = Nothing OrElse rblMajors.SelectedIndex = -1 OrElse txtDate.Text = Nothing OrElse txtDonation.Text = Nothing \_

 OrElse IsNumeric(Convert.ToDecimal(txtDonation.Text)) = False Then

 Response.Write("Check data entry")

 Exit Sub

 End If

 'if the program gets this far then the data entry was fine (no errors, no missing values). So we start putting the user-entered values into the columns for that new row. We named the new row dr for a shortcut name (dr = datarow)

 dr("CustomerName") = txtCustomer.Text

 dr("Major") = rblMajors.SelectedItem.Text

 'The FormatDateTime function is used to display the date selected in the calendar into a nicely formatted short date. To do this date formatting however we first pull the date selected out of the calendar using the dateTime.parse command. Next after pulling the date out of a textbox control, we can perform the formatting, using the formatdatetime(###, dateformat.shortdate) function.

 dr("TADate") = FormatDateTime(DateTime.Parse(txtDate.Text), DateFormat.ShortDate)

 dr("Donation") = Convert.ToDecimal(txtDonation.Text)

 'ok add the data row to the transactions table

 TransactionsTable.Rows.Add(dr)

 'here are some minor analytics for one column for all the rows. The table.compute is summing up the values in the entire donations column. Next we calculate the average for the donations column. Another time we introduce "Avg(Donation)" to average the column's values directly.

 gdecTotal = TransactionsTable.Compute("Sum(Donation)", Nothing)

 gdecAverage = gdecTotal / TransactionsTable.Rows.Count

 'ok show the analytics in a label.

 Label1.Text = "Total of " & TransactionsTable.Rows.Count & " transactions totalling " & gdecTotal.ToString("C0") & " The average donation amount is " & gdecAverage.ToString("C2")

 'now display the blue tabular datagrid control filled with data

 With TAGrid

 .DataSource = TransactionsTable

 .DataBind()

 .Visible = True

 End With

 'when adding a new row of data we clean up the screen clutter by removing the viewing of some gridviews. When entering data the program user wants to see the data. Another usage scenario is when the user wants to see analytics.

 AnalyticsGrid.Visible = False

 DeptGrid.Visible = False

 TAGrid.Visible = True

 End Sub

#End Region

#Region "Filter gridview to show only the transaction for the selected major"

 Protected Sub CheckBoxList1\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles CheckBoxList1.SelectedIndexChanged

 'if you are typing out this code, be sure to set the autopostback property of the checkbox list to true!

 'this code just filters the transactions to show only those donations for the selected major.

 Dim StrMajor As String

 Dim intCount As Integer

 'we introduce dataviews now. Dataview is a helper object for datatables. Dataviews are used to filter the rows to show from a datatable. (here the data structure that is holding the transaction records.

 'Similar programming objects are slicers in excel or PowerBI, or WHERE clauses in a SQL Statement.. A dataview is a helper object for the datatable that enables filtering. When the datatable is filtered to show a subset of the rows, then the gridview control just displays the rows that are viewable in the datatable. While here we filter on major, you can also filter on new rows, updated rows or recently deleted rows.

 'perform analytics, and filter the gridview control to show only transaction records for the department that is selected in the checkbox list. This select case is using the major that was selected in the check box list, and is passed into the server by the autppostback option.

 ' 'The hard part about this code is that the filter statement on the table.compute is a little touchy so we have to be careful to pass in a well formatted string of text. Notice this filter is also passed into the line of code that creates the dataview, so that the resulting display of transaction data is filtered on the department.

 Dim MajorsView As DataView

 Label3.Text = Nothing

 If TransactionsTable.Rows.Count = 0 Then

 Response.Write("Add data before Using this feature")

 Exit Sub

 End If

 If CheckBoxList1.SelectedIndex = -1 Then

 'if no filter is selected, then show all the transactions and exit this procedure.

 DeptGrid.Visible = False

 TAGrid.Visible = True

 Label3.Text = Nothing

 Exit Sub

 End If

 'this next code is tricky. The filter for the table.compute needs to read Datatable Column = 'Text filter'.

 'an example would be "Major = 'MIS' " The trick is to concatenate the single quotes around the major.

 'it would be nice if we could pass in the .selecteditem text from the checkbox list directly to the table.compute, but a string with single quotes around it is needed. Notice the red content before and after the checkbox.selecteditem.text it is really " ' " which adds a ' to the string variable.

 StrMajor = "Major = " & "'" & CheckBoxList1.SelectedItem.Text & "'"

 intCount = TransactionsTable.Compute("Count(TransactionID)", StrMajor)

 'if there are no donations yet for a major than stop the program, there is no data to show.

 If intCount = 0 Then

 Label3.Text = "No donations yet from " & CheckBoxList1.SelectedItem.Text

 Exit Sub

 End If

 'this is how you set up the dataview, no need to memorize this. We have to specify the table to filter, the filtering condition, the primary key column, and the data to filter.

 MajorsView = New DataView(TransactionsTable, StrMajor, "TransactionID", DataViewRowState.CurrentRows)

 'ok now display the filtered data.

 With DeptGrid

 .DataSource = MajorsView

 .DataBind()

 .Visible = True

 End With

 Label3.Text &= vbNewLine & vbNewLine & "Total for all " & CheckBoxList1.SelectedItem.Text & " is: " & vbTab & FormatCurrency(TransactionsTable.Compute("Sum(Donation)", StrMajor), 2)

 TAGrid.Visible = False

 End Sub

#End Region

#Region "Perform Analytics"

 Protected Sub Button6\_Click(sender As Object, e As EventArgs) Handles Button6.Click

 'This procedure performs and displays analytics by adding one row for each major to the analytics datable. The rows of donations data in the transactions table are compiled and condensed (which is the basis of analytics) and displayed in the analytics table. Condensed analytics for each major are calculated, and displayed as one row in the summary analytics datatable.

 'Here a datatable is used to store and display the results of analytics operations. The code presented here is one method to provide this functionality (others revealed in later programs). We create the rows of calculated condensed analytics in the datatable anew each time the button is pressed.

 'Obviously if there were millions of rows of transaction data to compile and perform analytics upon, this methodology presented here would be too slow. When you have big data its best to perform data condensing and analytics processing in the SQL Server database using perhaps a GROUP BY query which is built for that scale. But here you learn about data management and analytics in a lite infrastructure approach.

 'First clear out any rows of old analytics data

 AnalyticsDataTable.Rows.Clear()

 Dim strFilter As String

 'This code is dynamic in that if a new major was added to the radiobutton list then you wouldn't have to change a thing. The analytics would still work, give that a try! Here we run a loop that will run once for each item in the items collection of the radiobutton list (each major). Inside the loop a new row is created each time the tllo is run and calculated metrics are added to the columns for that row. Finally the row of analytics is added to the summary analytics datatable. You can check the names of the column in procedure above that created the datatables.

 For Each li As ListItem In rblMajors.Items

 Dim dr As DataRow = AnalyticsDataTable.NewRow

 'Now perform the analytics and record them. Write the major into the first column of the new row. li is a variable that allows you to iterate the items collection of a list control such as the radiobutton list. Li.text pulls the name that is displayed in the list control.

 dr("Major") = li.Text

 'this code is similarly tricky as explained above the concatenation is tricky. A single quote must be placed around the major. The table compute filtering statement requires a string, so this work was required. May not look like it, but this next line was the hardest in the entire program. Take a good notice of the use of single quotes and double quotes.

 'strFilter = "Major = " & " ' " & li.Text & " ' "

 strFilter = "Major = " & "'" & li.Text & "'"

 'here we calculate the total $ amount and number of donations for the current major being analzed. It is important to understand that the strfilter will be different

 dr("NumberDonations") = TransactionsTable.Compute("Count(TransactionID)", strFilter)

 dr("TotalDonations") = TransactionsTable.Compute("Sum(Donation)", strFilter)

 'the averages can now be calculated. This is tricky we have to first make sure we have some donations before we average them.

 Select Case dr("NumberDonations")

 Case 0

 'do nothing

 Case Is > 0

 'if there are rows of data to average then do it. This code demonstrates that you can derive data from other data. Here values from some columns that were already filled above are used to calculate values for other columns. Remember that the gdecTotal and gdecAverage are global variables that hold running totals for all the donations. We could have used a table.compute with nothing as the filter to recalculate these here.

 dr("%Total") = FormatPercent(dr("TotalDonations") / gdecTotal)

 dr("AvgDonation") = FormatNumber(dr("TotalDonations") / dr("NumberDonations"))

 dr("CompareToAvg") = FormatNumber(dr("AvgDonation") - gdecAverage)

 End Select

 'ok add that row of analytics to the datatable now

 AnalyticsDataTable.Rows.Add(dr)

 Next

 'ok now we can display the compiled analytics in the gridview control

 With AnalyticsGrid

 .DataSource = AnalyticsDataTable

 .DataBind()

 .Visible = True

 End With

 'and turn off the display of the detail transaction data.

 TAGrid.Visible = False

 End Sub

#End Region

#Region "Clear data entry controls"

 Protected Sub Button5\_Click(sender As Object, e As EventArgs) Handles Button5.Click

 txtCustomer.Text = Nothing

 txtDate.Text = Nothing

 txtDonation.Text = Nothing

 rblMajors.SelectedIndex = -1

 Label3.Text = Nothing

End Sub #End Region End Class