Using arrays for data management, metrics design and data visualization©

This project shows how to create data tables (ie arrays) for several purposes. Records form business transactions are stored into them, running totals for groups are kept in summary datatables, new columns of calculated metrics are added to summary data tables, and finally the summary data table is used as the datasource for tables and graphs.

Imports System.Data

'A prior program showed how to store rows of transaction data into an array. In a later separate set of procedures the data in the data structure was compiled into meaningful business performance measures to design and display common analytics. The funtionality of the program was purposefully segmented into two processes, 1) data validation and entry and 2) data analysis.

'This program again accepts new rows of data into a data structure (datatable), and then updates some columns in a second summary table (the classic SQL GROUP BY query or Excel pivottable, etc). This project uses in webpage memory data structures. This is a HUGE leap forward. Another version will demonstrate the usage of SQL and databases.

'This program replaces the use of global variables for running totals, to using of data structures which are more compact. So if your programs start to have too many variables for running totals, or your output has too much concatenation, consider using a data structure, which are cleaner looking. After some set-up and learning data structures are easier to work with and you will wonder why they were not introduced earlier. This program is used to teach data structures before moving on to web page <---> SQL Server database programs. Being able to build data structures, update them, and then use them for data visualizations is a core learning objective for the course, and these concepts are used in many different analytics/BI implementations.

'Pause for effect. While this project is similar to the "store data in Excel, use formulas/charts or pivot charts for data management/visualization there is an important upgrade and productivity gain resultant from using real programming and data structures. BTW why do so many managers cling to their Excel data and charts, when the processes can be automated? Perhaps fear and a feeling of lack of control. An upgrade over 'Excel databases' is that this type of implementation has data validation so the data being captured and stored is accurate.

'Again the scenario is that the girl scouts need to keep track of how many boxes of girl scout cookies they have sold. Sales are aggregated by troop and are compared to sales goals for each troop. The girl scout shows her sales sheet to the district leader whom uses this system to record how many boxes of cookies were sold, how much money was submitted, scout identification and troop membership.

'FYI - this chart control can only display numbers and words and cannot do any summarization/calculations. So we build and maintain a summary table of performance metrics. Witheach new transaction the summary table is updated. Another methodology would be to not have a summary table (which goes against the corporate mantra "I want my chart in 3 seconds or less") and rather derive the summary numbers from the transaction table ( ie fact table) when the chart procedure is run (table.compute or a SQL GROUP BY query are common).

'The best way to learn this important data structures, data management, data analysis, data visualization is to:

'1. Run this program over and over until you understand what functionality occurs with each event

'2. Print and read the code and commentary for one event procedure, then run that event procedure. Keep reading/running the procedure until it makes sense, refer to the glossary as needed to make sense of the terminology. Do not skip ahead until you understand how that darn code can do the magic.

'3. Make small changes to the program and watch the functionality change.

'4. Explain the program to a friend

'5. Make a new program of your own design (this is the no pressure fun learning experience)

'6. Work on an assigment to solidify your understanding. Demo the program to a friend, be able to explain to them the magic of coding.

'Note: The chart control seemed to work only with | FILE | New | Project ...not | File | New | Website.

Public Class GirlScoutCookies

 Inherits System.Web.UI.Page

 'we declare three data structures at a global web page level so that different proecdures can interract with the data. The first just shows the list of girl scout troops

 'The third data table is used to storerecords of the individual transactions which here is the collection of funds. This is similar to the fact table in the database

 Public Shared gtblTroopGoals, gtblTroopTotals, gtblRecordedSales As New DataTable 'this table used for the running totals

#Region "Initialize datatables"

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

 'Now we add the columns to the three datatables so this is essentially doing the same steps 3 different times. If this was a project connected to a database then the tables wold be made in the database software not in the memory of an ASP.NET webpage.

 'Lets's make sure this only occurs once! The next f the table already had columns then quit.

 If gtblTroopGoals.Columns.Count > 0 Then

 Exit Sub

 End If

 'here we add the columns, giving them a name and a data type.

 With gtblTroopGoals

 .Columns.Add("Troop", Type.GetType("System.String"))

 .Columns.Add("UnitsGoal", Type.GetType("System.Decimal"))

 End With

 If gtblTroopTotals.Columns.Count > 0 Then

 Exit Sub

 End If

 ' ------------------------Create the datatable for View #3

 'here we again add the columns, giving them a name and a data type. This is an upgrade fron using Excel which does not place restrictions on what you can type into a cell, which is the root of trouble with Excel data (people type unexpected values into columns).

 ' It is worth rememebring that if you want to perform an average (as shown below), you need to use the decimal datatype for the columns not integer (such as Int32 below). The metrics of total boxes, # transactions, avg. number boxes sold per scout, total revenue, percent of goal and troop sales status can all be calculated 'on-the-fly' by querying/compiling data from the transaction table and displayed on a managerial dashboard. That approach show using table.compute here and shown in another class. Here we keep these fields updated whenever a new sales transaction is made.

 ' Question: Why does the Percent of Goal column below string and not the numeric (decimal) data type?

 ' Answer: The string can be formatted with a percent sign..

 With gtblTroopTotals

 .Columns.Add("Troop", Type.GetType("System.String"))

 .Columns.Add("Boxes", Type.GetType("System.Decimal"))

 .Columns.Add("AvgBoxes", Type.GetType("System.Decimal"))

 .Columns.Add("NumberTA", Type.GetType("System.Decimal"))

 .Columns.Add("Revenue", Type.GetType("System.Decimal"))

 .Columns.Add("UnitsGoal", Type.GetType("System.Decimal"))

 .Columns.Add("Percent of Goal", Type.GetType("System.String"))

 .Columns.Add("Status of Sales for Troop", Type.GetType("System.String"))

 'these columns will store running totals so we need to inititalize their values, changing the value from NULL to 0. Doing so, allows the running totals to work later on. Remember that you can't add a number to a NULL cell, but can add a number to a 0 value. The goal of selling 50 boxes is the same for each troop, and is inserted into the UnitsGoal column for each new row. Same goals for each troop are a simplification, so a future iteration of the program can allow the goal to be entered by the program user. For example some troops are smaller and other larger in participation.

 .Columns("Boxes").DefaultValue = 0

 .Columns("AvgBoxes").DefaultValue = 0

 .Columns("NumberTA").DefaultValue = 0

 .Columns("Revenue").DefaultValue = 0

 .Columns("Percent of Goal").DefaultValue = Nothing

 .Columns("Status of Sales for Troop").DefaultValue = Nothing

 End With

 'The above datatable is used for the running totals. Pretty elaborate, but as you will see highly value added. With six columns automatically filled in when we add a new troop (as a new data row) we only have to add values to two columns when we create the new data row. Do you know them?

 ' ------------------------Create the datatable that is displayed on View #2

 'This procedure creates the data structure to store the data that is displayed in a gridview on View #2. The data table is created up at the page level just in case a different procedure needs to access the data structure (its common to hace one procedure add rows to the datatable, another procedure update the data structure, and a third to produce analytics. Again - this is a one-time set-up of the columns (their column names and their data types). The first column is even defined to be integer and auto-numbering, so the transaction # can increment as new datarows are added.

 'We again start with some error checking to see if the data table already has columns of data defined. That is a one-time process that runs only the first time the web page starts up. This next code makes sure that the columns are only added once.

 If gtblRecordedSales.Columns.Count > 0 Then

 Exit Sub

 End If

 'Here we add the columns to the table used to record the hundreds of girlscout cash receipts and recording the # of boxes of cookies sold. Notice the different data types (integer for whole numbers, string for words, decimal for bigger numbers). We have to add each column one at a time and specify the data type. the term after the WITH statement is added to the beginning of each line of code before the End With statement.

 With gtblRecordedSales.Columns

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

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

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

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

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

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

 End With

 'next we specify the first column to be auto-numbering starting at 1 and incrementing 1 each new record. Each sales transaction then is given a unique number so it can be referred to such as a purchase order #, or sales #, etc.

 With gtblRecordedSales.Columns("TransactionID")

 .AutoIncrement = True

 .AutoIncrementSeed = 1

 .AutoIncrementStep = 1

 End With

 End Sub

#End Region

#Region "Set up troop record - adding troop # and units goal"

 Protected Sub ImageButton1\_Click(sender As Object, e As ImageClickEventArgs) Handles ImageButton1.Click

 'This program keeps track of the cookie unit sales by troop and compares sales to a sales goal for each troop. When the program starts there are no troops in the system, so the troop numbers and their sales goal are captured and stored in a table first before any sales processing can occur. This procedure therefore continues the set-up process, first the tables are created and new rows are added to the troopstotals datatable, one new row is added for each troop. The district scout leader adds the troop numbers and their sales goal. A little magic also occurs, the radiobutton list of troops is also built in this procedure, so that later sales transaction processing occurs.

 'So here we add troops to the Troops Total Table. This would be a one-time task for each troop. Future versions of this software can include a way to edit this data and other troop master data (relatively unchanging data such as contact info) after it has been saved.

 'During program development and system testing, the web browser can remember the troops even after closing the browser so this next line clears out any data to start afresh.

 If rblTroops.Items.Count = 0 Then

 gtblTroopTotals.Rows.Clear()

 gtblRecordedSales.Rows.Clear()

 End If

 If txtTroop.Text = Nothing OrElse IsNumeric(txtTroop.Text) = False OrElse txtUnits.Text = Nothing \_

 OrElse IsNumeric(txtTroop.Text) = False Then

 txtOutput.Text = "Check data entry"

 Exit Sub

 End If

 'this next line adds the troop number to the items list of the radiobutton list control that is on View #2. The radiobutton list is used in sales transaction recording. So the program user gets to enter the troop numbers and their goals (but no way in this version to unwind a troop if the data is wrong). Advanced databased versions of this program would save the data into database tables, which can be updated.

 With rblTroops

 'this property can be set at design time, but wanted to show that properties can be set in code. Notice the new item being added to the radiobuttonlist is stored as text not an integer. This is so that later when the troop # is used as the X axis of a chart, that the chart is drawn correctly. For analytics the dimension being used to categorize the data usually needs to be text, so numbers and dates would need to be converted to text later on.

 .BorderColor = Drawing.Color.DodgerBlue

 .Items.Add(txtTroop.Text)

 End With

 'this is how you stuff the values from view#1 into a table structure for display to show the program user that their action was recorded.

 'this is an easy example there are only 2 columns (Troop and UnitsGoal)

 'So the steps are

 '1. create the new blank row of data for the data structure, passing in the schema so the columns are known and data typed (ie a numeric field expects a number to be entered).

 '2. populate the columns of that new row with data

 '3. add the 'filled-in' row of data to the data structure

 '4. display the table with the new rown in a gridview control

 Dim dr As DataRow = gtblTroopGoals.NewRow 'step 1

 With gtblTroopGoals

 dr("Troop") = txtTroop.Text

 dr("UnitsGoal") = Convert.ToDecimal(txtUnits.Text)

 End With 'step 2

 gtblTroopGoals.Rows.Add(dr) 'step 3

 'ok now show a gridview with the columns on the first view. Never forget that if you want to manipulate the data stored in teh data structure then manipulate the datatable. Leave the gridview alone, it only displays the data and does not store the data.

 GridView3.DataSource = gtblTroopGoals

 GridView3.DataBind() 'step 4

 'there is a more in-depth data structure on View #3. The TroopTotals table stores the running totals of cookie sales and provides some metrics related to progress towards goal. You can learn a lot from the usage of the data structure (aka matrix, array) and adding new columns of metrics. Here we generate new columns of metrics in VB.NET code inside the ASP.NET webpage. Another strategy is to use SQL code in a SQL query in your database to generate new columns of metrics. Featherman's analytics classes demonstrate this indistry-standard functionality. that project will use SQL Server database software, and specifically Microsoft's SQL Server Management Studio.

 ' -------------------put a new row into the data structure on View #3 --------------------------------------------

 Dim dr2 As DataRow = gtblTroopTotals.NewRow 'step 1

 'here are the initial values for each column of the new row being added to the data structure. Each row of the table represents one troop name and their running totals and current sales metrics as compared to sales goals. Four other columns have a default value of 0 so those columns will have a 0 recorded in them for this new row.

 dr2("Troop") = txtTroop.Text

 dr2("UnitsGoal") = Convert.ToDecimal(txtUnits.Text) 'step 2

 'Step 3 add a row to a data structure (datatable).

 gtblTroopTotals.Rows.Add(dr2)

 'ok now show a gridview with the columns

 GridView1.DataSource = gtblTroopTotals

 GridView1.DataBind()

 'and clear out the data entry controls

 txtTroop.Text = Nothing

 txtUnits.Text = Nothing

 End Sub

#End Region

View #2 Cash Receival System starts here ---------------------------------------------------



#Region "Save one row of transaction data"

 Protected Sub SaveDataRow()

 'When a new sale is recorded, a new row of data is added to the transaction table and the running totals for the troop are updated. The prior procedure was getting too long so this 'insert a row of data into the data structure' code was removed from the UDATE datatable code above. This INSERT new row procedure is called (and run) from the prior procedure. Breaking code into segments cleans up the program and makes it more manageable, kindof like cleaning and organizing your kitchen pantry.

 'We are recording the sales of boxes of cookies for one girl scout, so we are saving the sales data as a new row in the transactions table called gtblRecordedSales. This procedure adds a new row at the bottom of that data structure, fills each of it's columns with data and then save the row. Its important to remember that the new row of data expects the data types that were specified at table creation.

 If txtNumberBoxes.Text = Nothing OrElse txtCashCollected.Text = Nothing OrElse txtName.Text = Nothing OrElse IsNumeric(txtNumberBoxes.Text) = False OrElse IsNumeric(txtCashCollected.Text) = False OrElse rblTroops.SelectedIndex = -1 Then

 txtOutput.Text = "Check data entry"

 Exit Sub

 End If

 'create the new datarow and fill it with values

 Dim dr As DataRow = gtblRecordedSales.NewRow

 'here we populate each column of the new row added to the datatable. Remember the first column is auto-incrementing so that you do not have to insert the value, the value is added automagically.

 dr("Troop") = rblTroops.SelectedItem.Text

 dr("ScoutName") = txtName.Text

 dr("Boxes") = Convert.ToDecimal(txtNumberBoxes.Text)

 dr("Revenue") = Convert.ToDecimal(txtCashCollected.Text)

 dr("Date") = Now.ToShortDateString

 'the last line saves todays date in a short text format. Future programs can save the date using an actual date datatype to allow date and time metrics using timespan calculations. For example you could calculate the number of days since last sale recorded for any troup or scout.

 'now that the new data row is populated with values in each column, now add the row to the table and display the data in the gridview. Setting the gridview's datasource to the datatable again now that it it has a new row, will cause it to show the new data.

 gtblRecordedSales.Rows.Add(dr)

 GridView2.DataSource = gtblRecordedSales

 GridView2.DataBind()

 'give a little feedback that the sale was recorded

 txtOutput.Text = "sale recorded for troop " & rblTroops.SelectedItem.Text

 End Sub

#End Region

#Region "Sales transaction recording and update the running totals for the troop that the scout recorded sales for"

 Protected Sub Button1\_Click1(sender As Object, e As EventArgs) Handles Button1.Click

 Call SaveDataRow()

 'here is an improvement to the updating procedure - we wil lupdate the summary data for the 1 troop that a sale was recorder for

 Dim decBoxes, decNumberTA, decPercentofGoal As Decimal

 Dim strStatus As String

 'This procedure starts when the sales transaction is being recorded (see the first line of code Calls another procedure). Here updates are processed in the troops total table. The running total values for one girl scout troop are updated with the # of boxes sold and cash received. This procedure begins by calling another procedure that inserts the row of data to record the cash received into the transaction table then performs the update to the running totals table. So this procedure is doing data management with two data structures one at the highly granular transaction record level, another at the summary level (troop #).

 'now update the troop's totals for boxes and revenue in the running totals data table. How does the procedure know which row of data to update? The row # to be updated is in parentheses below (rblTroops.SelectedIndex). The rows in the radio button list and the troop totals table are both numbered with an index# that always start at 0. The first item in the radiobutton troops list is indexed item 0, the second is index item 1. So the row to be updated is catured in the radiobutton list based on where it is in the indexed list. This assumes that the row sorting is natural (based on what order it was created) and no subsequent resorting of rows in the radiobutton list and data table occur. This obscure problem handled in later SQL Server based programs.

 'Notice the += at the end of each of the next two lines, it means add the value from the textbox to the column values (Boxes, Revenue) that exist for the row of data that needs to be updated. Please recall that the Boxes and Revenue columns were given the default value of zero to ensure that running totals can be processed. Again you can't add a number to a NULL value in a datatable or SQL Server table, so the default value of the column must be set to zero.

 'this is so clever, we update the troop totals for the troop that was selected in the radiobutton list

 With gtblTroopTotals.Rows(rblTroops.SelectedIndex)

 .Item("Boxes") += Convert.ToDecimal(txtNumberBoxes.Text)

 .Item("Revenue") += Convert.ToDecimal(txtCashCollected.Text)

 .Item("NumberTA") += 1

 .Item("AvgBoxes") = FormatNumber(.Item("Boxes") / .Item("NumberTA"), 1) 'we have to format the # of decimal places the calculated value will have

 decPercentofGoal = .Item("Boxes") / .Item("UnitsGoal")

 Select Case decPercentofGoal

 Case Is <= 0.3

 strStatus = "plan to extend efforts"

 Case 0.301 To 0.667

 strStatus = "Making progress!"

 Case 0.6671 To 0.999

 strStatus = "Almost there!"

 Case 1 To 1.999

 strStatus = "Made it!"

 Case Is > 2

 strStatus = "Crushed it!!!"

 End Select

 .Item("Percent of Goal") = decPercentofGoal.ToString("P2")

 .Item("Status of Sales for Troop") = strStatus

 End With

 'The updates are completed so show the updated table in the gridview control

 GridView1.DataSource = gtblTroopTotals

 GridView1.DataBind()

 'with the running totals updated, call the procedure to insert the new datarow to record the transaction. Recall this is a bridge project where you learn ADO.NET concepts. Later projects will leverage this knowledge and introduce SQL databases, SQL programming language but don't worry, only for INSERT new row operations, and UPDATE running totals operations. Please remember that it is better to split the code into three procedures (for clarity, organization [separate regions], and ease of maintaining code later)

 End Sub

#End Region

View #3 ‘Metrics and Charting’ starts here ---------------------------------------------------

#Region "Another way to perform metrics - do it for all troops at the same time"

 'this procedure is not used in this project and is placed here to show what is possible. This project rather uses the procedure above to perform status and metrics update for ONE datarow, ONE troop. You might need to update an entire summary table, for example before refreshing charts, so the code below to demostrates that functionality.

 Protected Sub ComparisontoGoal()

 'The district leader needs to communicate the cookie sales progress to goal to each of the troop leaders, so that the troop leaders can formulate new sales strategies as needed (such as set up a table outside Martin stadium in Pullman on gameday). The troop leader needs feedback to make changes to their sales strategies. After each sale the progress to goal is re-calculated and displayed in the sales summary table. A common coding way to provide this functionality is to use a loop. This procedure uses a loop to recalcualte the metric for each row of data (for each troop).

 'We are fortunate to use a method we already understand to complete this work. Don't fall in love with loops however for this kind of functionality, better to do this work using database technologyuse SQL UPDATE SET code (saved into a stored procedure that can be called as needed), if the data is in a SQL Server database (covered elsewhere). The UPDATE SET command can have a CASE statement and is less lines of code.

 'The code is in a loop so that all the rows of the data structure are updated. The code demonstrates a common methodology, copy values from a few columns for a row into local variables, then perform some calculations, and finally write back some values into the same and/or different columns for that same row, and proceed to the next row. The processing can be quite elaborate, and it is common to run routines against each row in a dimension table such as each employee, vendor, or store. The code inside loops can call functions written by other developers. Here we update the % of sales goal attained, and status. The data management concepts revealed here are just as important as the actual programming skills and syntax shown.

 Dim strStatus As String

 Dim decPercentofGoal As Decimal

 Dim intBoxes, intGoal, intTA As Integer

 For Each dr As DataRow In gtblTroopTotals.Rows 'see we run this code for every row in the totals table. We calculate some metrics.

 'pull values form the current row into local variables so you can do further processing

 intBoxes = dr("Boxes")

 intGoal = dr("UnitsGoal")

 intTA = dr("NumberTA")

 If intBoxes > 0 AndAlso intGoal > 0 AndAlso intTA > 0 Then

 decPercentofGoal = intBoxes / intGoal

 'we will display a different text message in the status column for the current row, so we use a SELECT CASE to apply the differential text based on progress to goal. The term will be stored in the column for the current row.

 Select Case decPercentofGoal

 Case Is <= 0.3

 strStatus = "Plan to extend efforts"

 Case 0.301 To 0.667

 strStatus = "Making progress"

 Case 0.6671 To 0.9999

 strStatus = "Almost there"

 Case 1 To 1.999

 strStatus = "Made it!!"

 Case Is >= 2

 strStatus = "Crushed it!!"

 End Select

 'these next lines update two of the columns in the datatable for the current row. Remember that the code is being run once for each row in the datatable. We are still inside the loop. First the calculated % Percent of Goal attained is copied into the cell; then the status is written to the column for the current row (ie "Made it").

 'The else statement works in conjunction with the IF statement above which makes sure that there are values in each of the columns needed for the calculations. If there are NOT the requisite values then the loop skips to the next row for possible calculations there. This is the usage of Continue FOR meaning continue the FOR NEXT loop if there is a data problem, skip ahead to the next row in the datatable.

 dr("Percent of Goal") = decPercentofGoal.ToString("P2")

 dr("Status of Sales for Troop") = strStatus

 Else : Continue For

 End If

 Next

 End Sub

#End Region

#Region "Clear form and link buttons"

 Protected Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

 Call Clear()

 End Sub

 Private Sub Clear()

 'clear out the web form. This procedure is put into a procedure that can be called from different procedures.

 txtName.Text = Nothing

 txtCashCollected.Text = Nothing

 txtNumberBoxes.Text = Nothing

 txtOutput.Text = Nothing

 txtTroop.Text = Nothing

 txtUnits.Text = Nothing

 rblTroops.SelectedIndex = -1

 End Sub

 'these linkbuttons allow you to navigate amongst the three views

 Private Sub LinkButton1\_Click(sender As Object, e As EventArgs) Handles LinkButton1.Click

 MultiView1.ActiveViewIndex = 0

 End Sub

 Private Sub LinkButton2\_Click(sender As Object, e As EventArgs) Handles LinkButton2.Click

 MultiView1.ActiveViewIndex = 1

 End Sub

 Private Sub LinkButton3\_Click(sender As Object, e As EventArgs) Handles LinkButton3.Click

 'When the button is clicked for the third view the chart is updated

 MultiView1.ActiveViewIndex = 2

 DrawChart()

 End Sub

#End Region

#Region "Charts"

 Private Sub DrawChart()

 'this procedure draws two charts. The datasources are set, titles are specified,

 ' the x and y axis are specified, sometimes the charts are specified to be 3D, and the type of chart is specified.

 With Chart1

 .DataSource = gtblTroopTotals

 .DataBind()

 .Titles.Add("Total # Boxes Sold by Troop")

 End With

 'set the titles

 With Chart1.ChartAreas(0)

 .AxisX.Title = "Troops"

 .AxisY.Title = "# Boxes for Troop"

 .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

 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. PowerBI or Tableau ahve charts that run a GROUP BY () query behind the scenes to do the aggregation work. The chart control is cannot run calculations, they only display the compiled numbers that are in the datatable.

 ' The X and y data values come from the datatable. There are many chart types (change the last property of the next line to see them).

 .ChartType = DataVisualization.Charting.SeriesChartType.Column

 .XValueMember = "Troop"

 .YValueMembers = "Boxes"

 .IsValueShownAsLabel = True

 End With

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

 With Chart2

 .DataSource = gtblTroopTotals

 .DataBind()

 .Titles.Add("Total # Boxes Sold by Troop")

 .Legends.Add("Legend1") 'the pie chart looks better with a legend

 .Legends("Legend1").Enabled = True

 .ChartAreas(0).Area3DStyle.Enable3D = True

 End With

 'set the data columns to show in the chart

 With Chart2.Series("Series1")

 .ChartType = DataVisualization.Charting.SeriesChartType.Pie

 .XValueMember = "Troop"

 .YValueMembers = "Boxes"

 .IsValueShownAsLabel = True

 .IsVisibleInLegend = True

 End With

 'we will do more with PowerBI and SSRS for reporting, those are fully-featured chart tools.

 End Sub

#End Region

End Class