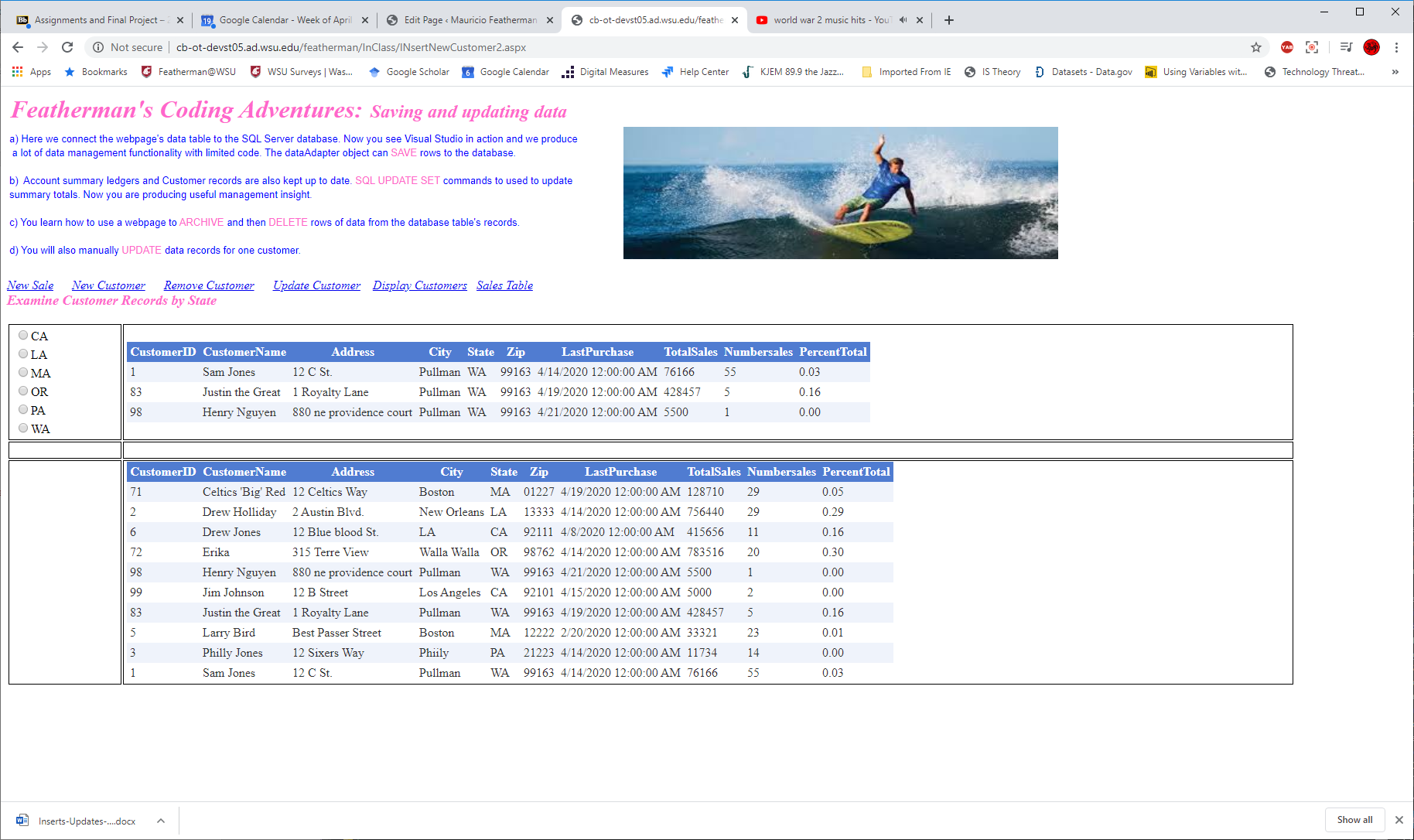
Featherman’s Coding Adventures©: More data management techniques  
  
Imports System.Data

Imports System.Data.SqlClient

'This program demonstrates inserts of new rows of data from the webpage to the connected SQL Server databasae table. A row of data is added to a dimension table (new customer) and transaction table (new sale). Customers are also updated or deleted from the database. The insert and update paradigm is also demonstrated where several columns of measures in the customer dimension table is updated automagically after a new sale is made. ALso show is how to delete rows from a child fact (sales) table so that you can delete the customer form the dimension (customers) table.

Partial Class InClass\_INsertNewCustomer2

Inherits System.Web.UI.Page

#Region "Connection"

Public Shared con As New SqlConnection(Put your connection string here)

#End Region

'these ADO.NET objects are used to select, retrieve, display and update the customer list in both dropdownlists and gridview. Two type of SELECT statements are used the first retrieves all the rows from the specified table. The second type of SELECT query we run is called parameterized becuase we need to pass it a value to filter the retrieved data by (e.g. state, customerID).

'If you want To display the complete table Of data Or list Of data (such As all the customers here) it must not be parameterized. Use Of a commandbuilder tells you that table inserts will be performed. Be careful, while global datatables are very useful to store and make accessible data for processing, they also need to constantly be refreshed so that the correct number of rows are displayed. Refreshes are common,

Public Shared gdaCustomers As New SqlDataAdapter("SELECT \* FROM Customers2020 ORDER BY CustomerName", con)

Public Shared cbCustomers As New SqlCommandBuilder(gdaCustomers)

Public Shared gCustomersTable, gCustforDDLTable As New DataTable

'A separate data adapter is needed to connect to, select and insert rows into each database table. This type of coding then is repetetive. This next set of ADO.NET objects is used to interract with the sales table. Because this next SQL SELECT statement reads to select the TOP 1 record from a table, it is used to display the sale data after it is saved.

Public Shared gdaSales As New SqlDataAdapter("SELECT Top 1 \* FROM Sales2020 ORDER BY SaleID DESC", con)

Public Shared cbSales As New SqlCommandBuilder(gdaSales)

Public Shared gSalesTable As New DataTable

'We use a separate set of ADO.NET objects to retrieve and update ONE customer record. Again the commandbuilder is the unsung hero that will be used to insert and update data in this table.

Public Shared gOneCustomerTable As New DataTable

Public Shared gdaGet1Customer As New SqlDataAdapter("SELECT \* FROM Customers2020 WHERE CustomerID = @p1", con)

Public Shared CommandBuilderGet1 As New SqlCommandBuilder(gdaGet1Customer)

Public Shared gdecNumberCustomers, gdecNumberSales, gdecTotalRevenue As Decimal

Public Shared gArchivesTable As New DataTable

#Region "Code to run before the page is ready to be used: Page Init"

Protected Sub Page\_Init(sender As Object, e As EventArgs) Handles Me.Init

'We will use two in-memory data tables in this program, they are assigned schema here. These two datatables will connect to corresponding SQL Server databse tables, to save/retrieve data from the permanent database. You connect the database table to the corresponding datatable array. In this procedure in one line of .Fillschema code we copy in the column schema information to the datatable (this replaces the columns.Add code we wrote previously.

'show the list of customers on view 1 when the program starts up

Call getCustomers()

'fill three different dropdown lists with the customer name and ID

Call UpdateDDL()

End Sub

#End Region

#Region "Load DDL’s and the Customer Gridviews"

Private Sub getCustomers()

'At different times in the program we want to refresh the customers list that is displayed, for example to update a custpomer's totals after a successful transaction. This is a good example of a sub procedure reused over and cover and called from other procedures that need help. After you insert, update, or delete a customer you want to see that that change was completed right? So here first we clear out the global datatable and gridview then fill it again.

If gCustomersTable.Rows.Count > 0 Then gCustomersTable.Rows.Clear()

GridView1.DataSource = Nothing

GridView1.DataBind()

Try 'now we get the newest version of the Customers data to display in on the webpage.

gdaCustomers.Fill(gCustomersTable)

GridView1.DataSource = gCustomersTable

GridView7.DataSource = gCustomersTable

GridView1.DataBind()

GridView7.DataBind() 'this gridview is on the Display Customers page

'why not give the program user some value? OF course they need summary numbers at the top of a list. The Table compute is an easy tool to use in webpages.

gdecNumberCustomers = gCustomersTable.Rows.Count

gdecTotalRevenue = gCustomersTable.Compute("SUM(TotalSales)", Nothing)

gdecNumberSales = gCustomersTable.Compute("SUM(NumberSales)", Nothing)

TextBox9.Text = "Current totals " & gdecNumberCustomers & " customers, " & gdecNumberSales & " transactions totalling " & gdecTotalRevenue.ToString("C0") & " average sales transaction was " & FormatCurrency(gdecTotalRevenue / gdecNumberSales, 0)

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

#End Region

#Region "More Page Refresh Code here refresh the DDL's (dropdownlists)"

Private Sub UpdateDDL()

'this code refreshes the list of customers shown in the dropdown lists. This is repetitive code that actually needs to be run frequently from different places within this program. Rather than write duplicate code into different procedure (making it harder to maintain), its better to place repetitive code in a separate sub procedre that gets 'called' to run from within other processes.

'This UpdateDDL procedure is run when the webpage is initialized, when a new customer is added, when the data for a customer is edited, and when a customer is deleted.

If gCustforDDLTable.Rows.Count > 0 Then gCustforDDLTable.Rows.Clear()

Try 'these dropdown lists are on different views

gdaCustomers.Fill(gCustforDDLTable)

With DropDownList1

.DataSource = gCustforDDLTable

.DataTextField = "CustomerName"

.DataValueField = "CustomerID"

.DataBind()

.Items.Insert(0, "Select a customer")

End With

With DropDownList2

.DataSource = gCustforDDLTable

.DataTextField = "CustomerName"

.DataValueField = "CustomerID"

.DataBind()

.Items.Insert(0, "Select a customer")

End With

With DropDownList3

.DataSource = gCustforDDLTable

.DataTextField = "CustomerName"

.DataValueField = "CustomerID"

.DataBind()

.Items.Insert(0, "Select a customer")

End With

Catch ex As Exception

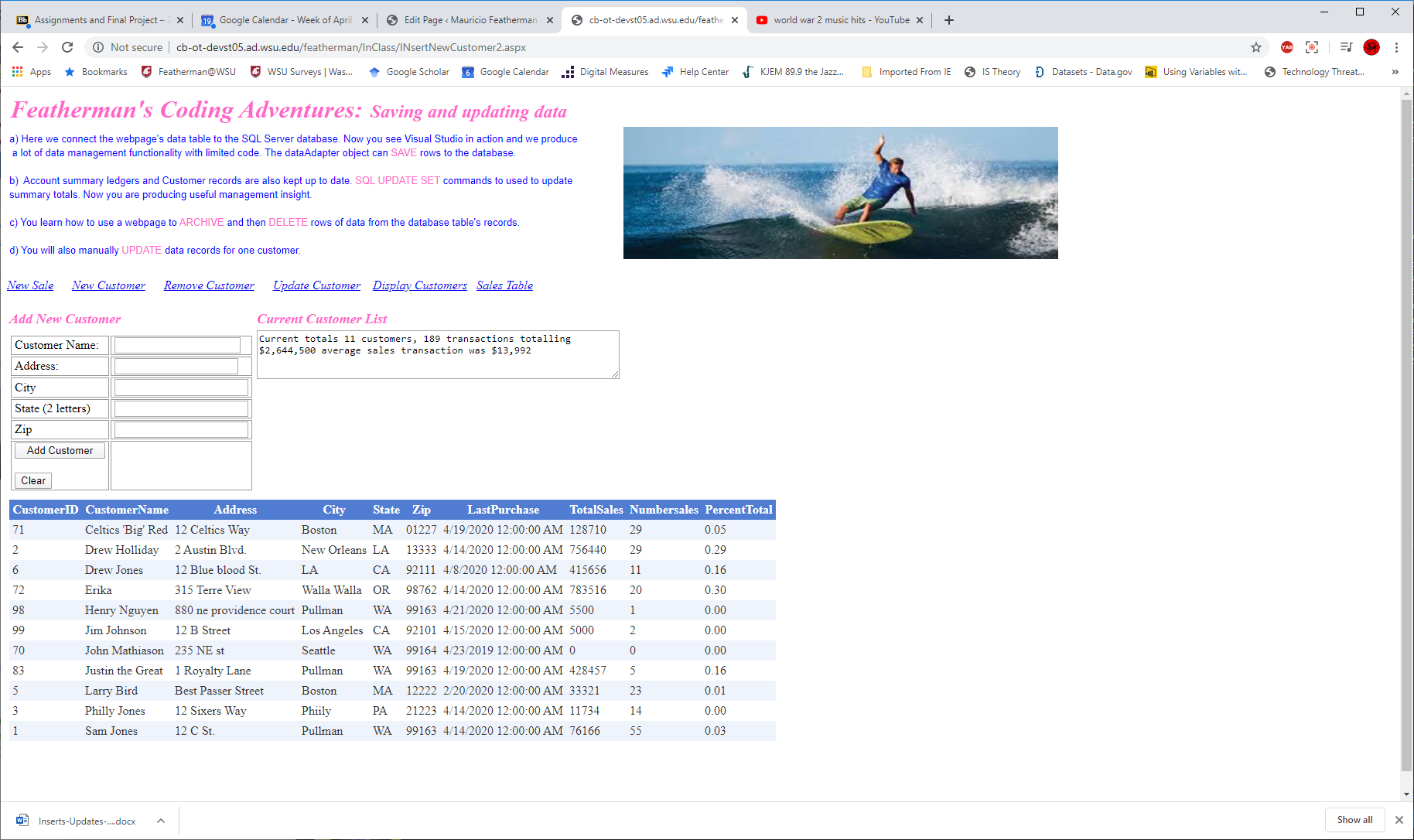
Response.Write(ex.Message)

End Try

End Sub

#End Region

#Region "View #2: Add New Customer"

 Protected Sub Button7\_Click(sender As Object, e As EventArgs) Handles Button7.Click

'This webpage records sales to pre-existing customers. Anonymous sales are not supported so customers must be registered first. This procedure then, is used to add a new customer to the customers database table. We first make sure that customer is not already in the database table, to protect against someone pressing the save button twice. This first data adapter checks to see if there is a row of data with the same customer name. If no row is found, then the customer data is saved into the SQL database.

'Here we copy the schema into the in-memory ASP.NET datatables on the webpage so that the add New customer is a super easy dataAdapter.update. The In-memory array now has a schema (the datatable knows its columns) the list Of columns, their datatypes and restrictions, keys, default values, autonumbering etc.

gdaCustomers.FillSchema(gCustomersTable, SchemaType.Mapped)

Dim CheckCustomerTable As New DataTable

Dim daCheckCustomer As New SqlDataAdapter("SELECT \* From Customers2020 WHERE CustomerName = @p1", con)

With daCheckCustomer.SelectCommand.Parameters

.Clear() 'clear out any prior values

' to improve the search capability of the parameter we trim off any trailing spaces in the textbox after the customers name

.AddWithValue("@p1", TextBox1.Text.Trim)

End With

If TextBox1.Text = Nothing OrElse TextBox2.Text = Nothing OrElse TextBox3.Text = Nothing OrElse TextBox4.Text = Nothing OrElse TextBox5.Text = Nothing Then

Response.Write("Enter data")

Exit Sub

End If

Try 'this next line of code runs the red SQL SELECT statement above to talk to the SQL Server database.

daCheckCustomer.Fill(CheckCustomerTable)

'We don't want to add the same customer more than once so we check the database table to see if there already exists a customer with the same name. The SELECT query will return a row of data if the customer name is already in the database. So it is simple to count how many rows were returned from the database. If there is a row of data returned then we stop data processing and exit the procedure. We do not want to add the same customer twice.

If CheckCustomerTable.Rows.Count >= 1 Then

TextBox9.Visible = True 'run this code only if the customer name is already found

TextBox9.Text = TextBox1.Text & " is already a member of the clubhouse"

Exit Sub

End If

'this code only runs if the customer name was not found in the database table. Here we create a new row with the schema of the customer database table. We create the new row and fill all of its columns except the primary key field (customerID) which is autonumbering.

Dim dr As DataRow = gCustomersTable.NewRow

dr.Item("CustomerName") = TextBox1.Text

dr.Item("Address") = TextBox2.Text

dr.Item("City") = TextBox3.Text.Trim

dr.Item("State") = TextBox4.Text

dr.Item("Zip") = TextBox5.Text

dr.Item("LastPurchase") = Today.Date

'These last two columns will keep running totals and therefore need to start with a zero value.

dr.Item("TotalSales") = 0

dr.Item("Numbersales") = 0

gCustomersTable.Rows.Add(dr)

'now push the new row of data that is in the webpage's memory back to the SQL Server database

gdaCustomers.Update(gCustomersTable) 'this runs the insert record SQL code because the dataAdapter is helped out by its good friend the commandbuilder (which wrote the parameterized SQL insert statement for us

'update the customers gridview on view 1, to show the new customer

Call getCustomers()

'update the dropdown lists on each view because to show this new customer

Call UpdateDDL()

Catch ex As Exception

Response.Write(ex.Message)

Finally

con.Close()

End Try

'remove the error messagecfrom view,

TextBox9.Visible = False

End Sub

#End Region

#Region "View#4: Get one customer record and update their data"

'To add more functionality to this demoware webpage, the capability to retrieve one customer then edit them is again covered

Protected Sub DropDownList3\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles DropDownList3.SelectedIndexChanged

 'The gdaGet1Customer dataAdapter and related commandbuilder that are utilized here are created as public shared objects at the page level (check the red sql code to see what table is being accessed. These ADO.NET objects and the datatable are created at the page level because the updating is performed in 2 separate procedures:

'1. The first procedure retrieves one row of data and is parsed and into the controls on the webpage. So one row of customer data is selected and displayed in the web controls.

'2. The second procedure takes the values from the webpage controls and puts them back into the row of data, and then pushed back to the database - in effect editing the values in the database.

'Because the process is a) download and display, b) wait for the program user to make their changes, c) push the changes back to the database, then you have to break the processing into 2 procedures. You can't start a procedure then make it wait for the program user. So here we only retrieve the row to update.

If DropDownList3.SelectedIndex <= 0 Then Exit Sub

With gdaGet1Customer.SelectCommand.Parameters

.Clear()

.AddWithValue("@p1", DropDownList3.SelectedValue)

End With

Try

'Since the datatable is global, we better make sure it is cleared out of prior data.

If gOneCustomerTable.Rows.Count > 0 Then gOneCustomerTable.Rows.Clear()

'This is where the one row of data is retrieved from the SQL Server database and copied into the datatable

gdaGet1Customer.Fill(gOneCustomerTable)

'Here we have row 0 and can access the data in the columns, passing those values to the webpage controls. The code below reads from right to left when there is an = sign. Recall the way to refer to one column in a one row of a datatable: table.rows(row #).item("columnName")

With gOneCustomerTable.Rows(0)

txtCN.Text = .Item("CustomerName")

txtA.Text = .Item("Address")

txtC.Text = .Item("City")

txtS.Text = .Item("State")

txtZ.Text = .Item("Zip")

txtLP.Text = .Item("LastPurchase")

txtTS.Text = .Item("TotalSales")

txtNS.Text = .Item("Numbersales")

End With

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

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

'This is the procedure to take the values from the webpage and place them ever so carefully into the columns for that row of data and then push the entire row of data back to the SQL server database table, replacing the old values for the row (here a customer) with the new values just retrieved from the webpage.

'Here since the assignment code reads from right to left, we take the values in the textboxes and assign it to the relevant columns for the row of data. It is not recommended to use dropdownlists, or textboxes that look like calendars, or calendar controls themselves. You can use a checkbox though.

With gOneCustomerTable.Rows(0)

.Item("CustomerName") = txtCN.Text

.Item("Address") = txtA.Text

.Item("City") = txtC.Text

.Item("State") = txtS.Text

.Item("Zip") = txtZ.Text

.Item("LastPurchase") = txtLP.Text

.Item("TotalSales") = txtTS.Text

.Item("Numbersales") = txtNS.Text

End With

'Lights, camera, action! Here the row of data is in the gOneCustomerTable data table, we use the dataAdapter's most excellent .Update function to take the data and push it back to the SQL Server database table. The unsung hero here is the SQL Commandbuilder which silently wrote the parameterized UPDATE SET command for us.

Try

gdaGet1Customer.Update(gOneCustomerTable)

'As a convenience here we show the one row of data that was saved back to the database

GridView4.DataSource = gOneCustomerTable

GridView4.DataBind()

'Here we update the customers gridview on View #1

getCustomers()

UpdateDDL()

MultiView1.ActiveViewIndex = 3

Catch ex As Exception

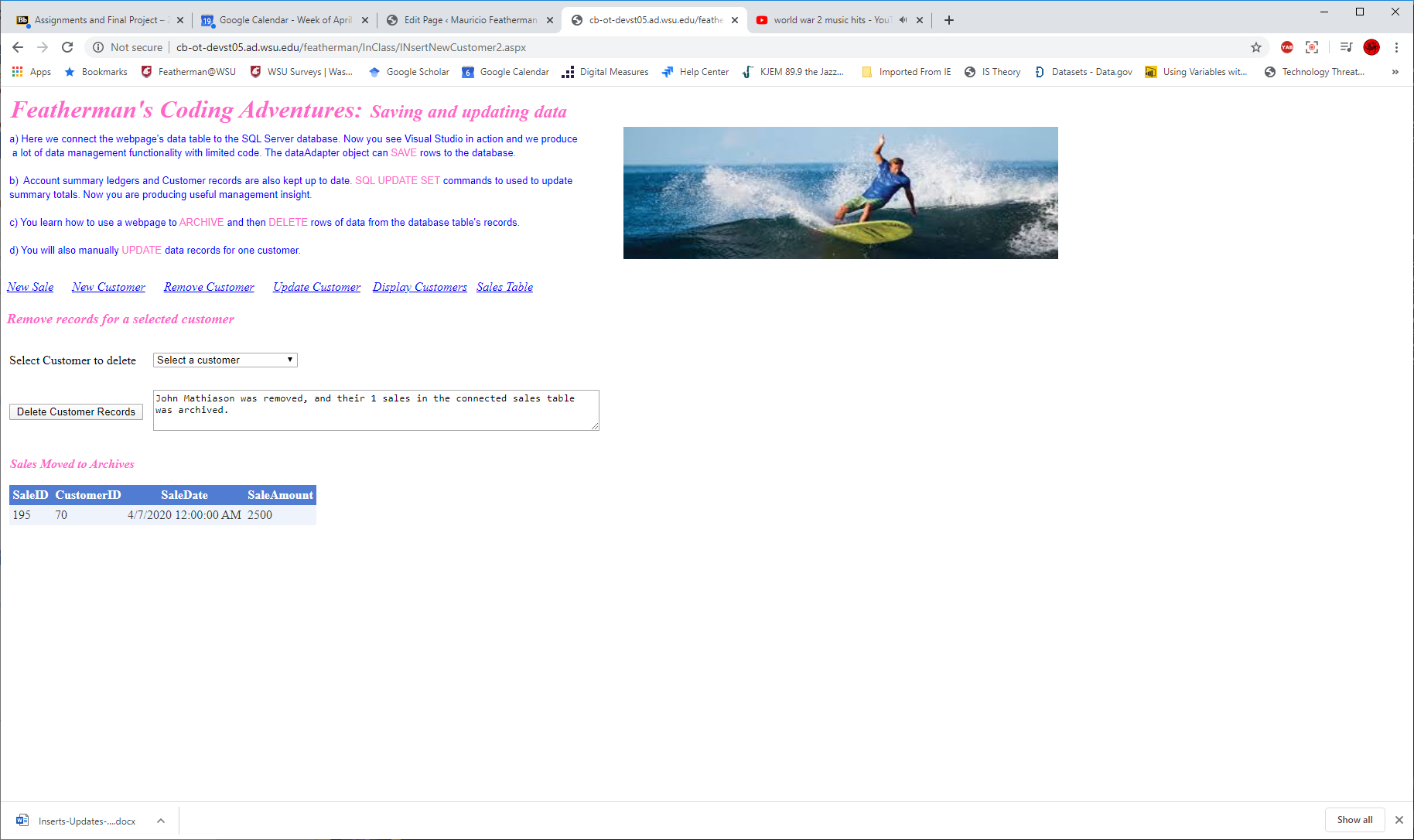
Response.Write(ex.Message)

End Try

End Sub

#End Region

#Region "View #3: Delete customer record"

 'There is a foreign key relationship between Customers2020 (the parent dimension table) and Sales2020 (the child or fact table of transactions). So customers have many sales. While you can delete a sale at any point (from the 'child' sales table), you cannot delete a row from the parent Customer table until you delete the related rows from the child Sales table (so we handle that here). This procedure demonstrates the simplicity of deleting rows of data. While here we just delete sales, in business systems you would probably move the sales records to a different 'archives' history table.

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

'Here we DELETE a customer record. IF they have any sales, those sales are archived.

Dim intSales As Integer 'used to show how many sales the customer had

'Here we will intercept the delete request and check if the customer has any sales, by counting the number of sales.

'If it is discovered that there are zero sales for the customer, then sure delete the customer. Otherwise,

'if it is discovered that the customer has sales then a message is shown to the program user, that it is not possible to

'delete the customer at this time (this program does not have a 'move to archives' procedure.

If DropDownList2.SelectedIndex <= 0 Then Exit Sub

Dim cmdCountSalesForOneCustomer As New SqlCommand("SELECT COUNT(\*) From Sales2020 WHERE CustomerID = @p1", con)

With cmdCountSalesForOneCustomer.Parameters

.Clear()

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

End With

'this next SQLcommand copies rows from the Sales table INTO the Sales Archives table.

Dim cmdSaveSalesToArchives As New SqlCommand("INSERT INTO Sales2020Archives SELECT \* From Sales2020 WHERE CustomerID = @p1", con)

With cmdSaveSalesToArchives.Parameters 'only copy the sales for that specific customer

.Clear()

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

End With

Dim DaShowArchives As New SqlDataAdapter("SELECT \* FROM Sales2020Archives WHERE CustomerID = @p1", con)

With DaShowArchives.SelectCommand.Parameters

.Clear()

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

End With

'These next two commands delete rows from the Customers and Sales tables.

Dim cmdDeleteCustomer As New SqlCommand("DELETE From Customers2020 WHERE CustomerID = @p1", con)

Dim cmdDeleteSalesFirstB4DeleteCustomer As New SqlCommand("DELETE From Sales2020 WHERE CustomerID = @p1", con)

'The dataAdapter will fetch rows of data from the Sales table for the selected CustomerID

With cmdCountSalesForOneCustomer.Parameters

.Clear()

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

End With

'this is the parameter for the Sqlcommand that will delete a customer from the customers table

With cmdDeleteCustomer.Parameters

.Clear()

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

End With

'this is the parameter for the Sqlcommand that will delete the sales for the customer that will be removed. When the tables are connected using a foreign key | primary key relationship you have to delete the related transaction records before you can delete a row from a dimension table (here customers).

With cmdDeleteSalesFirstB4DeleteCustomer.Parameters

.Clear()

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

End With

'All the code above was set-up. Here finally we fetch the sales for the customer selected in the dropdown list.

'see if the customer has any sales. If they do then we have to delete the sales before we delete the customer record

Try

If con.State = ConnectionState.Closed Then con.Open()

intSales = cmdCountSalesForOneCustomer.ExecuteScalar

If intSales >= 1 Then

'if the customer had sales, save them to archives, which is a second table with the same schema as Sales.

cmdSaveSalesToArchives.ExecuteNonQuery()

'Next display the records that are moved to archives, so clear the global table and then fill the gridview

If gArchivesTable.Rows.Count > 0 Then gArchivesTable.Rows.Clear()

DaShowArchives.Fill(gArchivesTable)

gvArchives.DataSource = gArchivesTable

gvArchives.DataBind()

'after the sales are archived then delete the sales for the selected customer, and then delete the customer record

cmdDeleteSalesFirstB4DeleteCustomer.ExecuteNonQuery()

cmdDeleteCustomer.ExecuteNonQuery()

txtOutput.Text = DropDownList2.SelectedItem.Text & " was removed, and their " & intSales & " sales in the connected sales table was archived."

Else

'This section of code only runs if there are zero sales records. So there are no sales to archive.

If con.State = ConnectionState.Closed Then con.Open()

cmdDeleteCustomer.ExecuteNonQuery() 'now we delete the customer record.

txtOutput.Text = "Records for " & DropDownList2.SelectedItem.Text & " removed. They had " & gSalesTable.Rows.Count & " sales."

End If

'After deleting a customer we need to refresh the dropdown lists to show that customer is no longer on the list

UpdateDDL()

'Also after deleting a customer you need to refresh the customer list in the gridview

Call getCustomers()

Catch ex As Exception

Response.Write(ex.Message)

Finally

con.Close()

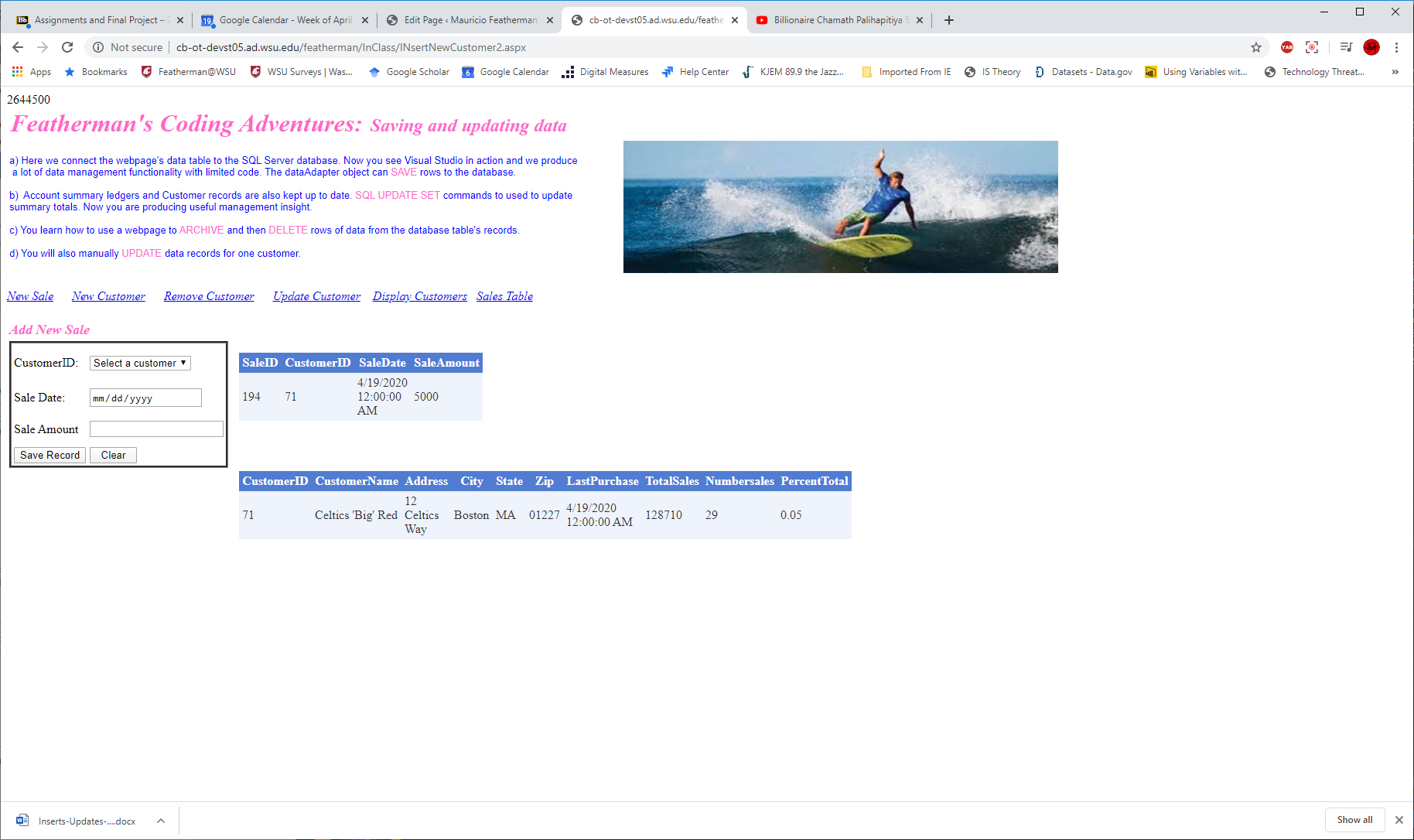
End Try

End Sub

#End Region

#Region "View #1: Save new sale for existing customer"

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

 'This procedure performs the insert and update metaphor, where a new sale is saved and some columns of a hybrid dimension table are updated. A hybrind dimension table holds the relatively unchanging master data for a record (here name & address of customer) and some columns of measures using the RFM approach. The measures are running totals for each customer including

'a) a measure of frequency - the number of transactions,

'b) a measure of recency - the date of the last time a sale was made to the customer

'c) a measure of total revenue generated by the customer.

'Before we add the row to the database we need to set up another dataAdapter so we can see the customer's updated totals after the new sale. After we insert the new sale, and update the one customer's master data record (update the running totals) we will want to show the updated customer totals by retrieving just the one row of customer data. This next dataAdapter and datatable will display just one customer row of data.

'First we map in the schema into the in-memory ASP.NET datatables on the webpage so that the add new sale is a super easy dataAdapter.update. The In-memory array now has a schema (the datatable knows its columns) the list Of columns, their datatypes And restrictions, keys, default values, autonumbering etc.

gdaSales.FillSchema(gSalesTable, SchemaType.Mapped)

Dim getUpdatedCustomer As New SqlDataAdapter("SELECT \* FROM Customers2020 WHERE CustomerID = @p1", con)

Dim oneCustomerTable As New DataTable

With getUpdatedCustomer.SelectCommand.Parameters

.Clear()

.AddWithValue("@p1", DropDownList1.SelectedValue)

End With

Dim dtSaleDate As Date 'we use a date variable to expand the formatting

'not a lot of data validation here but on this next line we see if a data was selected in the textbox. The IsDate() function is similar to the IsNumeric() and IsString() functions. If a date was selected then assign it to the date variable.

If DropDownList1.SelectedIndex <= 0 OrElse TextBox7.Text = Nothing OrElse TextBox8.Text = Nothing OrElse IsDate(DateTime.Parse(TextBox7.Text)) = False Then

Response.Write("Enter data for sale")

Exit Sub

End If

dtSaleDate = DateTime.Parse(TextBox7.Text)

'here is the new row of data for the sales table, now we fill the columns for the new blank row

Dim dr As DataRow = gSalesTable.NewRow

'This code is great. Becuase this procedure is called from the prior procedure, we can access the control values on the form.

'We do not have to use gloabl variables to pass values from the save sales record procedure to the update customer procedure.

dr.Item("CustomerID") = CInt(DropDownList1.SelectedValue)

dr.Item("SaleDate") = dtSaleDate.Date

dr.Item("SaleAmount") = Convert.ToDecimal(TextBox8.Text)

'here we add the new row of data to the datatable so that it can be pushed back to the database.

gSalesTable.Rows.Add(dr)

Try

'next run the insert (save) sales record, this is made very easy by our good friend the SQLcommandbuilder

gdaSales.Update(gSalesTable)

'clear the sales table and fetch the newly updated sales table data

gSalesTable.Rows.Clear()

GridView2.DataSource = Nothing

GridView2.DataBind()

gdaSales.Fill(gSalesTable)

GridView2.DataSource = gSalesTable

GridView2.DataBind()

'next call the sub procedure that will run the SQL UPDATE SET code that updates one row of customer data

Call UpdateCustomer()

Call UpdateCustomerPercents()

Call getCustomers()

'clear out the one row of customer data and display the updated customer record for the customer that just completed the sale.

If oneCustomerTable.Rows.Count > 0 Then oneCustomerTable.Rows.Clear()

getUpdatedCustomer.Fill(oneCustomerTable)

GridView3.DataSource = oneCustomerTable

GridView3.DataBind()

'update the gridview of customers on a different view - the customer update view

Catch ex As Exception

Response.Write(ex.Message)

End Try

'clear out the controls on the view

DropDownList1.SelectedIndex = -1

TextBox7.Text = Nothing

TextBox8.Text = Nothing

End Sub

'There is a clear button on the New Sale view, its code is next

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

GridView3.DataSource = Nothing

GridView3.DataBind()

GridView2.DataSource = Nothing

GridView2.DataBind()

End Sub

#End Region

#Region "Update Customer Metrics"

'now to update the customer running totals after the sales transaction. This separate called procedure segments and organizes code. It is often a good idea to segment and compartmentalize code into modules that can be called from within other procedures.

Private Sub UpdateCustomer()

'we use a SQLCommand (rather than a dataAdapter) to perform updates and deletes, to maximize your SQL learning.

If DropDownList1.SelectedIndex <= 0 Then Exit Sub

Dim cmdUpdateCustomer As New SqlCommand("UPDATE featherman.Customers2020 SET Numbersales += 1, TotalSales += @p1, LastPurchase = @p2 WHERE CustomerID = @p3", con)

'the parameters here allow the values from the webpage to be inserted into the red SQL statement above, when the .executenonquery function is run.

With cmdUpdateCustomer.Parameters

.Clear()

.AddWithValue("@p1", Convert.ToDecimal(TextBox8.Text))

.AddWithValue("@p2", DateTime.Parse(TextBox7.Text))

.AddWithValue("@p3", CInt(DropDownList1.SelectedValue))

End With

'All the above code was set-up. Next we run the red SQL code updating the row in the customer table. Then we call the getCustomers procedure to refresh all the data tables in the web page.

Try

If con.State = ConnectionState.Closed Then con.Open()

cmdUpdateCustomer.ExecuteNonQuery()

Catch ex As Exception

Response.Write(ex.Message)

Finally

con.Close()

End Try

End Sub

Protected Sub UpdateCustomerPercents()

Dim cmdGetTotal As New SqlCommand("SELECT SUM(TotalSales) FROM Customers2020", con)

'The following SQL syntax was necessary because computers do not like divide by zero errors. A percent of total metric is added to display each customer's sales as a percent of the total sales. A more advanced version of this calculation would be to filter sales by the last year to filter for recent business transactions. The UPDATE SET command calculates the row percentage total for each and every row in the Customers2020 table. A case statement is used which is just like an IF statement, but can enable more flexibility.

'The TotalSales/@p1 formula would crash if either the numerator or denominator were zeros. The case statement handles the Customers that have zero sales by assigning a 0 in that column.

Dim cmdUpdatePercents As New SqlCommand("UPDATE Customers2020 SET PercentTotal = (CASE WHEN TotalSales = 0 THEN 0 WHEN TotalSales > 0 THEN TotalSales/@p1 END) ", con)

Dim decTotal As Decimal

'Get value to pass as parameter into SQL query to be used as denominator of calculation

Try

If con.State = ConnectionState.Closed Then con.Open()

decTotal = cmdGetTotal.ExecuteScalar

Response.Write(decTotal)

Catch ex As Exception

Response.Write(ex.Message)

End Try

With cmdUpdatePercents.Parameters

.Clear()

.AddWithValue("@p1", decTotal)

End With

Try

If con.State = ConnectionState.Closed Then con.Open()

cmdUpdatePercents.ExecuteNonQuery()

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

#End Region

#Region "Utilities - clearing webform controls and linkbutton navigation to different views"

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

TextBox1.Text = Nothing

TextBox2.Text = Nothing

TextBox3.Text = Nothing

TextBox4.Text = Nothing

TextBox5.Text = Nothing

TextBox9.Text = Nothing

End Sub

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

MultiView1.ActiveViewIndex = 0

'clear out the last sale and customer record from the gridviews

GridView2.DataSource = Nothing

GridView2.DataBind()

GridView3.DataSource = Nothing

GridView3.DataBind()

End Sub

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

'beforeshowing this view, clear out/clean up the screen controls

MultiView1.ActiveViewIndex = 2

txtOutput.Text = Nothing

DropDownList2.SelectedIndex = -1

If gArchivesTable.Rows.Count > 0 Then gArchivesTable.Rows.Clear()

gvArchives.DataSource = Nothing

gvArchives.DataBind()

End Sub

Protected Sub LinkButton4\_Click(sender As Object, e As EventArgs) Handles lnkUpdateCustomer.Click

'Here when this linkbutton is selected we clear out any values from a prior transaction

MultiView1.ActiveViewIndex = 3

DropDownList3.SelectedIndex = -1

txtCN.Text = Nothing

txtA.Text = Nothing

txtC.Text = Nothing

txtS.Text = Nothing

txtZ.Text = Nothing

txtLP.Text = Nothing

txtTS.Text = Nothing

txtNS.Text = Nothing

End Sub

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

MultiView1.ActiveViewIndex = 1

getCustomers()

End Sub

#End Region

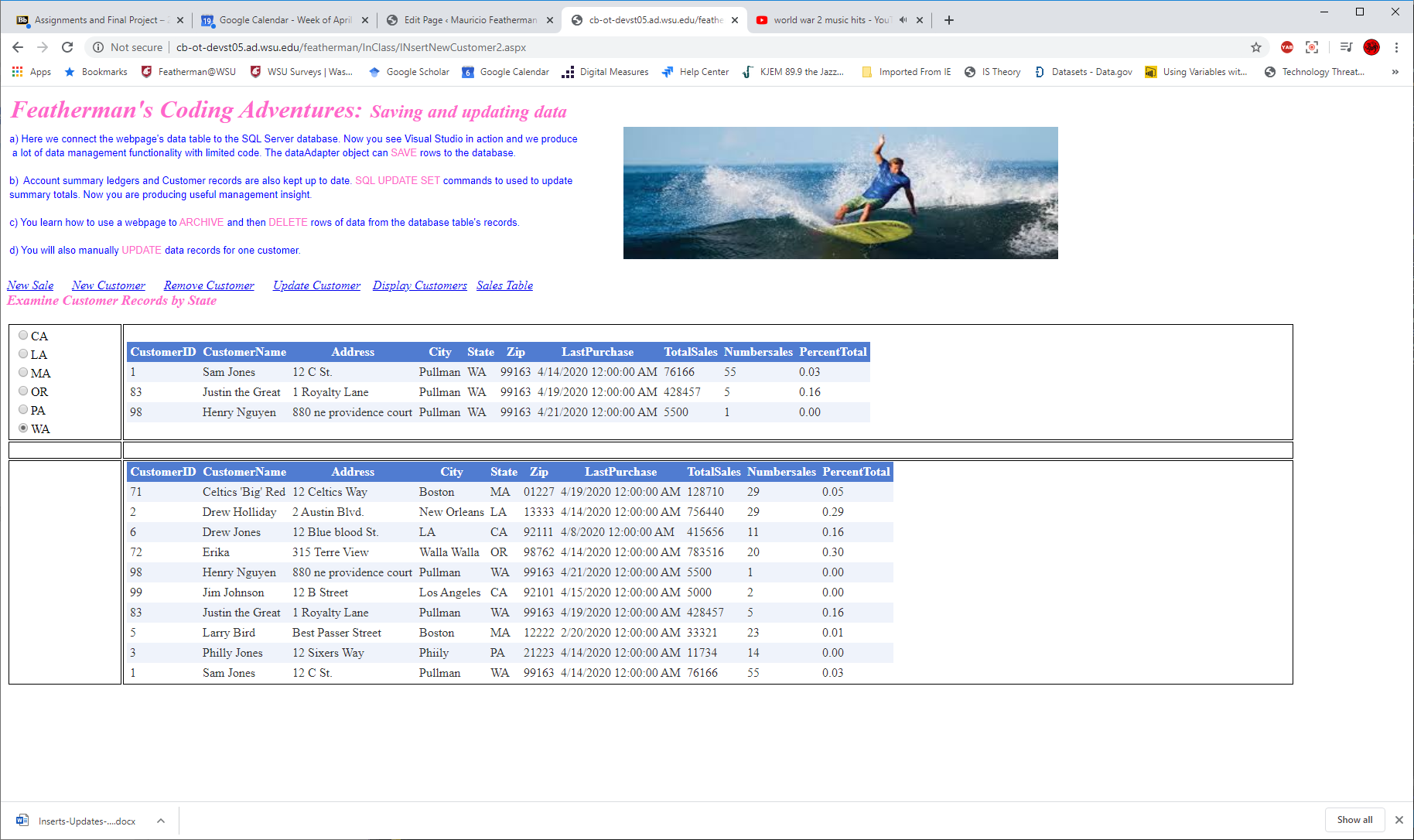
#Region "View #5: Examining customers"

Protected Sub LinkButton5\_Click(sender As Object, e As EventArgs) Handles lnkDisplay.Click

MultiView1.ActiveViewIndex = 4

'When the Display customers linkbutton is selected we need to fill the radiobuttonlist on the view with a list of the states. To remove duplicate states we use the term SELECT DISTINCT. Because we are using a local datatable it is created anew each time the procedure is run, therefore unlike a global datatable, we do not have to clear its rows to ensure we do not get duplicate values.

Dim daGetStates As New SqlDataAdapter("Select DISTINCT State FROM Customers2020", con)

Dim statestable As New DataTable

Try

daGetStates.Fill(statestable)

rblStates.DataSource = statestable

rblStates.DataTextField = "State"

rblStates.DataBind()

Catch ex As Exception

Response.Write(ex.Message)

End Try

getCustomers()

End Sub

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

MultiView1.ActiveViewIndex = 4 ' make sure we display the correct view

If rblStates.SelectedIndex = -1 Then Exit Sub

'This procedure fetches and displays the customers for the selected state in a radiobutton list

Dim CustomersForOneStateTable As New DataTable

Dim daGetCustomersForOneState As New SqlDataAdapter("SELECT \* FROM Customers2020 WHERE State = @p1", con)

'When you make a selection in the states list the state name is passed into the SQL SELECT code using this variable called @p1, parameter 1.

daGetCustomersForOneState.SelectCommand.Parameters.Clear()

daGetCustomersForOneState.SelectCommand.Parameters.AddWithValue("@p1", rblStates.SelectedItem.Text)

Try

daGetCustomersForOneState.Fill(CustomersForOneStateTable)

GridView5.DataSource = CustomersForOneStateTable

GridView5.DataBind()

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

#End Region

#Region "View #6: Show sales table"

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

MultiView1.ActiveViewIndex = 5

Dim SalesTable As New DataTable

Dim daGeSales As New SqlDataAdapter("SELECT SaleID, CustomerID, FORMAT(SaleDate, 'MM/dd/yyyy') as [Order Date], Format(SaleAmount, 'N0') as [Sales Total] FROM sales2020", con)

Try

daGeSales.Fill(SalesTable)

GridView6.DataSource = SalesTable

GridView6.DataBind()

Catch ex As Exception

Response.Write(ex.Message)

End Try

End Sub

#End Region

End Class