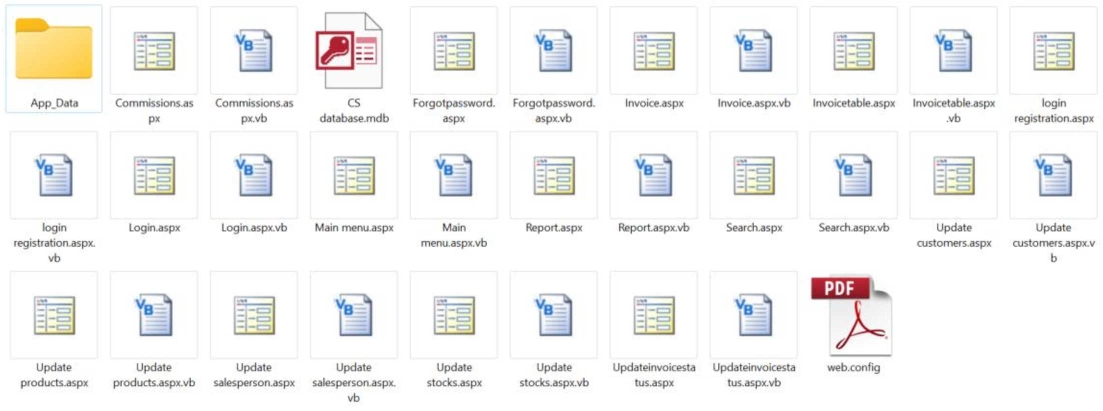
Criterion C: Development

Program Structure



Techniques used:

Table of Contents

1. Login Page

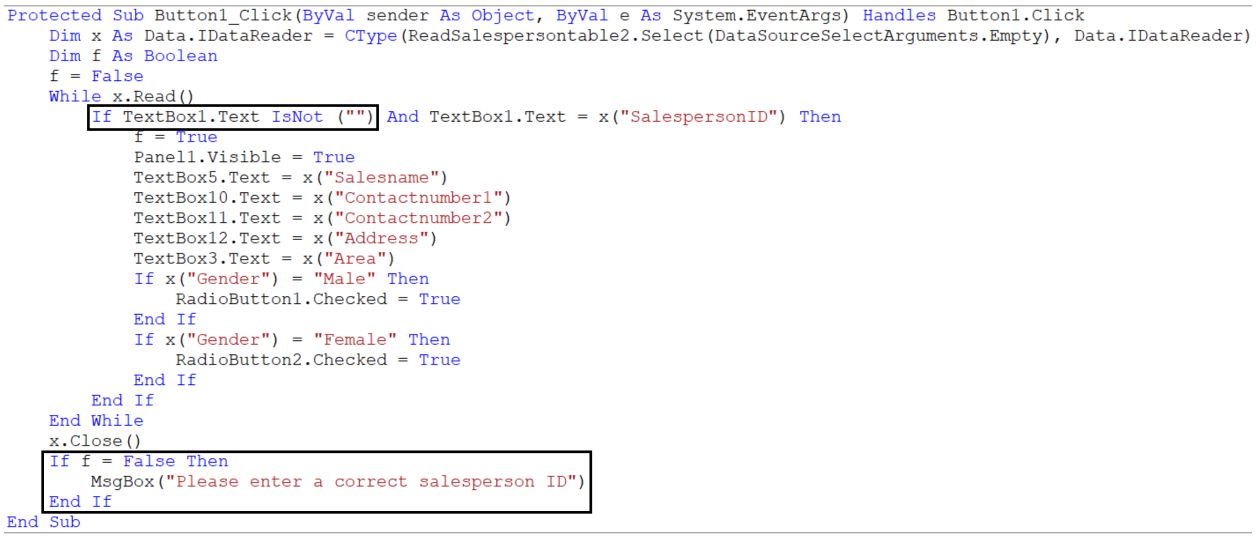
The login page is designed to require user to input their usernames and passwords and matching them with the registered data in the database. Only if both username and password match the registered data, the login page will redirect user to the main page, or else it will output an error message. The login page provides security for the program and prevent unauthorized users from accessing the data.

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    'To check whether inputted username and password are correct or matching with the database
    Dim x As Data.IDataReader = CType(Readlogin.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean = False
    While x.Read()
        If TextBox1.Text = x("Uname") And TextBox2.Text = x("Pword") Then
            Session("loginname") = x("Uname") 'session to store user's username to enhance user interface
            f = True
        End If
    End While
    If f = True Then
        MsgBox("You have logged in successfully")
        Response.Redirect("Main menu.aspx")
    Else
        MsgBox("Incorrect username or password")
    End If
End Sub
```

Figure 1.1: Login code for authentication

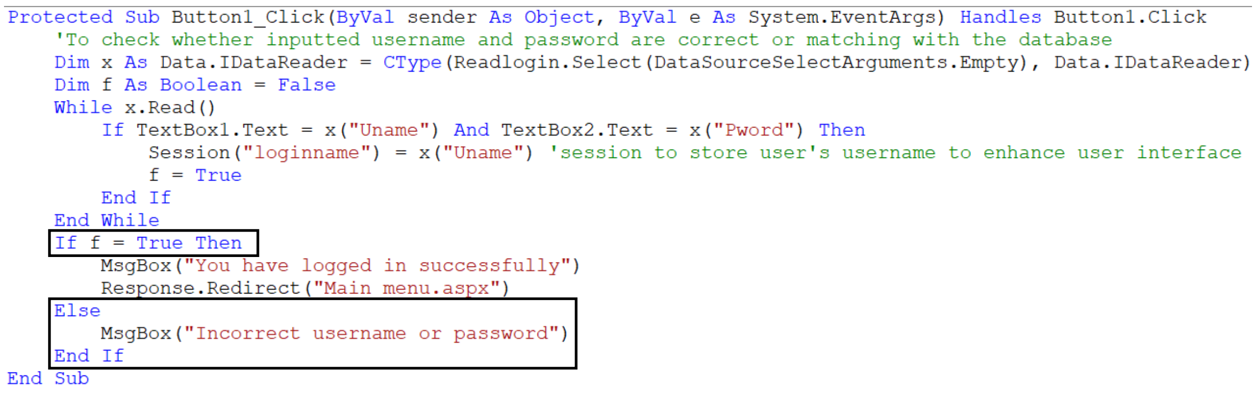
2. If else then condition

I use if else then condition throughout the program for various reasons such as for authenticating and validations. It helps specify what to do when the given criteria applied is valid or not. In validation, it will not allow empty boxes to be stored and become redundant data (use if condition to check). In authentication, if else then is used to check if the user is authorized if not, it will not grant access.



```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim x As Data.IDataReader = CType(ReadSalespersontable2.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean
    f = False
    While x.Read()
        If TextBox1.Text IsNot ("") And TextBox1.Text = x("SalespersonID") Then
            f = True
            Panel1.Visible = True
            TextBox5.Text = x("Salesname")
            TextBox10.Text = x("Contactnumber1")
            TextBox11.Text = x("Contactnumber2")
            TextBox12.Text = x("Address")
            TextBox3.Text = x("Area")
            If x("Gender") = "Male" Then
                RadioButton1.Checked = True
            End If
            If x("Gender") = "Female" Then
                RadioButton2.Checked = True
            End If
        End If
    End While
    x.Close()
    If f = False Then
        MsgBox("Please enter a correct salesperson ID")
    End If
End Sub
```

Figure 2.1: If-else Function to validate (present check) and return error message

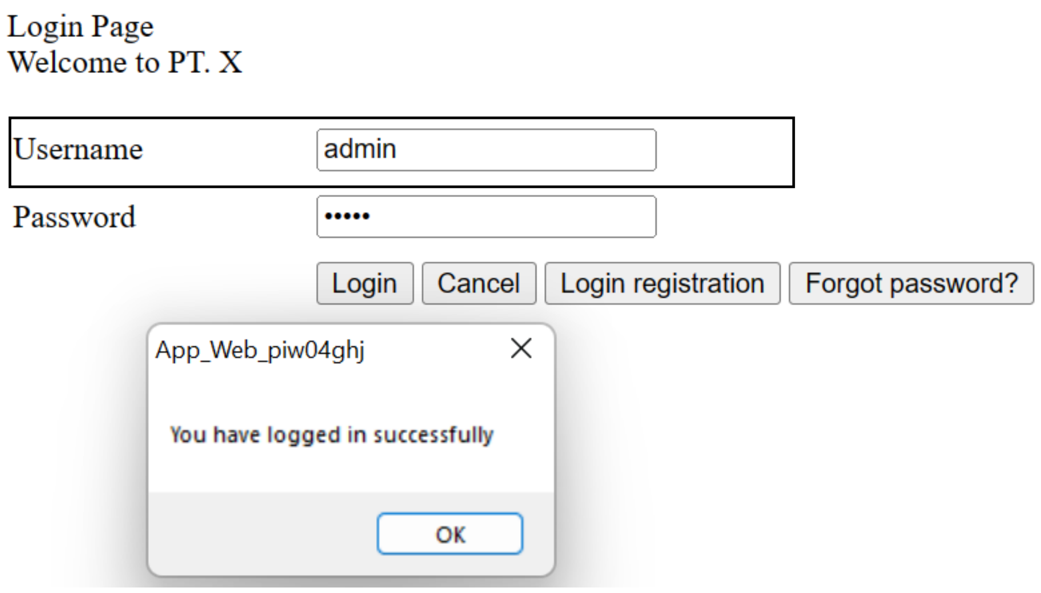


```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    'To check whether inputted username and password are correct or matching with the database
    Dim x As Data.IDataReader = CType(Readlogin.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean = False
    While x.Read()
        If TextBox1.Text = x("Uname") And TextBox2.Text = x("Pword") Then
            Session("loginname") = x("Uname") 'session to store user's username to enhance user interface
            f = True
        End If
    End While
    If f = True Then
        MsgBox("You have logged in successfully")
        Response.Redirect("Main menu.aspx")
    Else
        MsgBox("Incorrect username or password")
    End If
End Sub
```

Figure 2.2: If-Else function to provide error message in login page

3. Use of sessions

In the login page, I utilize sessions which act like cookies to save the entered username only if the user identity is authenticated. The saved username will then be displayed using a label in every page the user hover to therefore, makes user interface easier to use and easier tracking of the user identity.


Login Page
Welcome to PT. X

Username	admin
Password	•••••

You have logged in successfully

Figure 3.1: Login page to save user's username

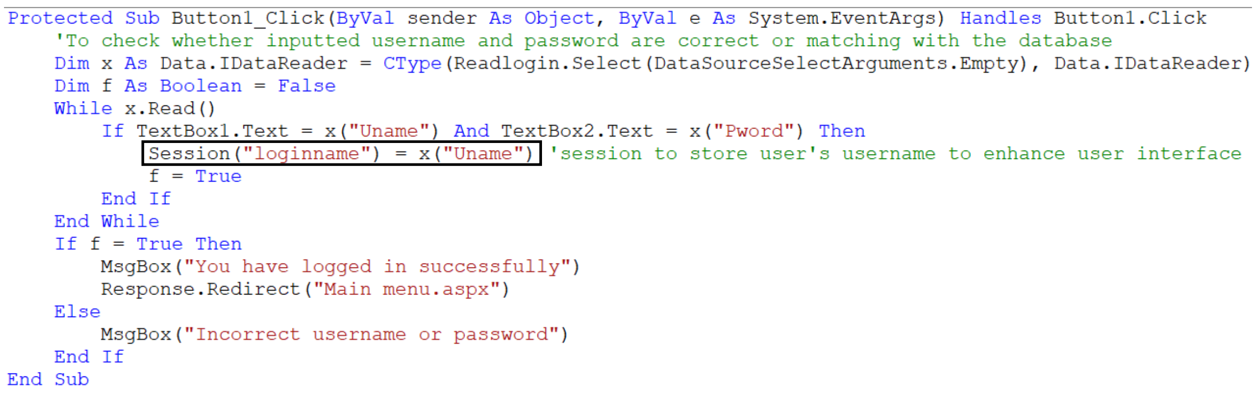
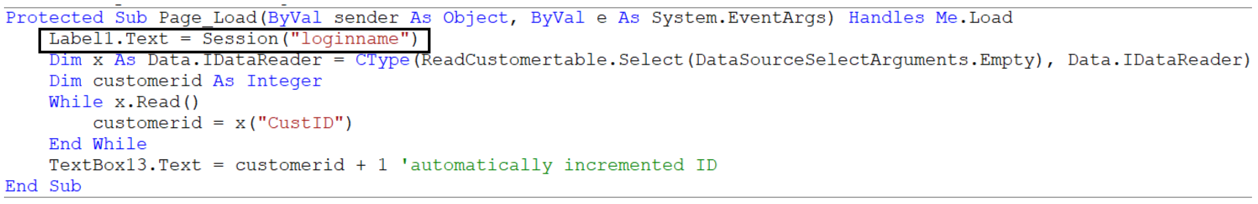

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    'To check whether inputted username and password are correct or matching with the database
    Dim x As Data.IDataReader = CType(Readlogin.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean = False
    While x.Read()
        If TextBox1.Text = x("Uname") And TextBox2.Text = x("Pword") Then
            Session("loginname") = x("Uname") 'session to store user's username to enhance user interface
            f = True
        End If
    End While
    If f = True Then
        MsgBox("You have logged in successfully")
        Response.Redirect("Main menu.aspx")
    Else
        MsgBox("Incorrect username or password")
    End If
End Sub
```

Figure 3.2: Session to store the username matched with the database record



```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Label1.Text = Session("loginname")
    Dim x As Data.IDataReader = CType(ReadCustomertable.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim customerid As Integer
    While x.Read()
        customerid = x("CustID")
    End While
    TextBox13.Text = customerid + 1 'automatically incremented ID
End Sub
```

Figure 3.3: Displaying the session through the usage of labels in every menu page

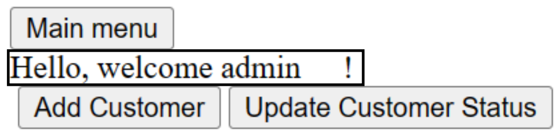
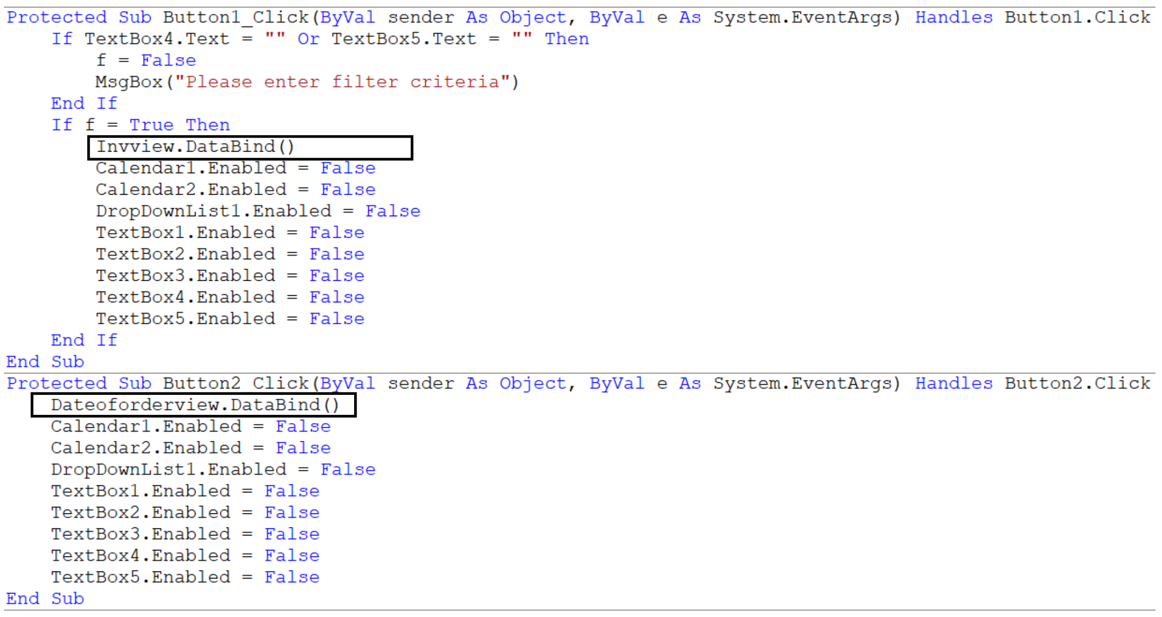


Figure 3.4: Session working successfully

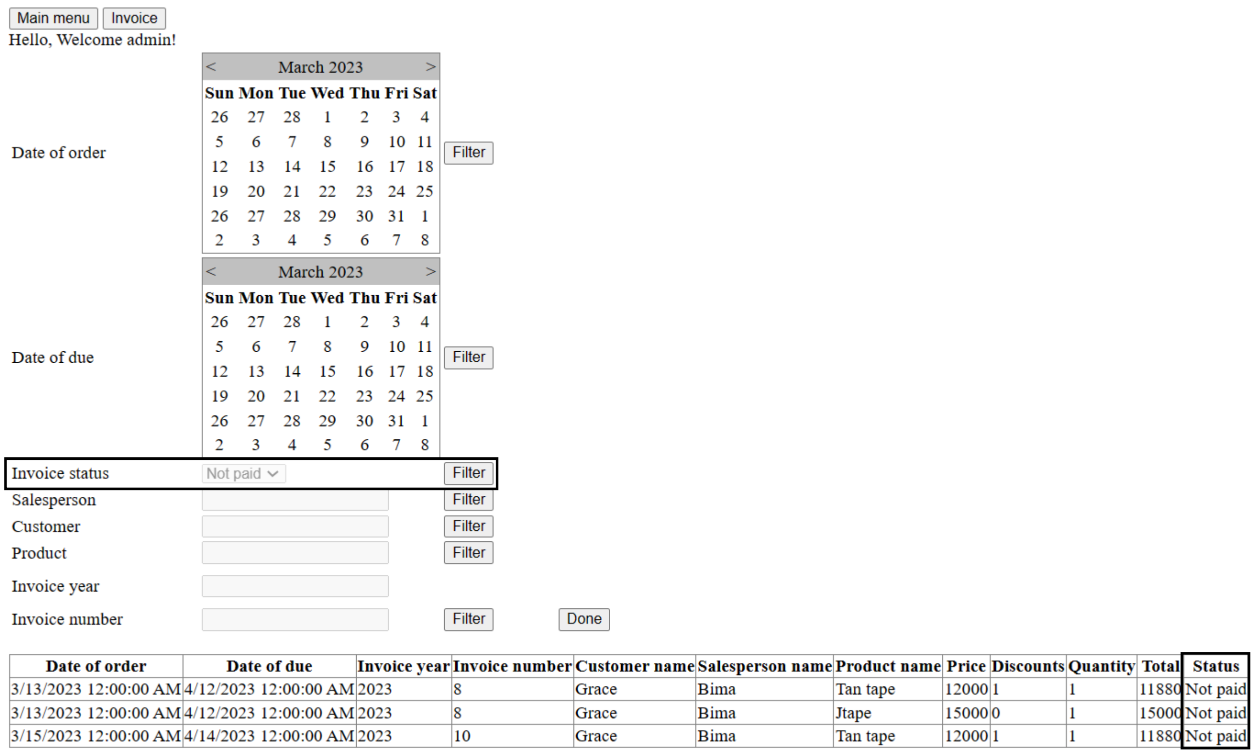
4. Use of data binding

A grid view can only read the user’s input once the data is bound. Therefore, I implement the technique of data binding to attach the set filter from the user to the browse feature in the complex query of the grid view. Hence, once the grid view reads from the data bound from user inputs and compare them against the database, it will show the filtered output.



```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    If TextBox4.Text = "" Or TextBox5.Text = "" Then
        f = False
        MsgBox("Please enter filter criteria")
    End If
    If f = True Then
        Invview.DataBind()
        Calendar1.Enabled = False
        Calendar2.Enabled = False
        DropDownList1.Enabled = False
        TextBox1.Enabled = False
        TextBox2.Enabled = False
        TextBox3.Enabled = False
        TextBox4.Enabled = False
        TextBox5.Enabled = False
    End If
End Sub
Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dateoforderview.DataBind()
    Calendar1.Enabled = False
    Calendar2.Enabled = False
    DropDownList1.Enabled = False
    TextBox1.Enabled = False
    TextBox2.Enabled = False
    TextBox3.Enabled = False
    TextBox4.Enabled = False
    TextBox5.Enabled = False
End Sub
```

Figure 4.1: Data binding code



Hello, Welcome admin!

Date of order

<			March 2023			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Filter

Date of due

<			March 2023			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Filter

Invoice status	Not paid	Filter
Salesperson		Filter
Customer		Filter
Product		Filter
Invoice year		
Invoice number		Filter

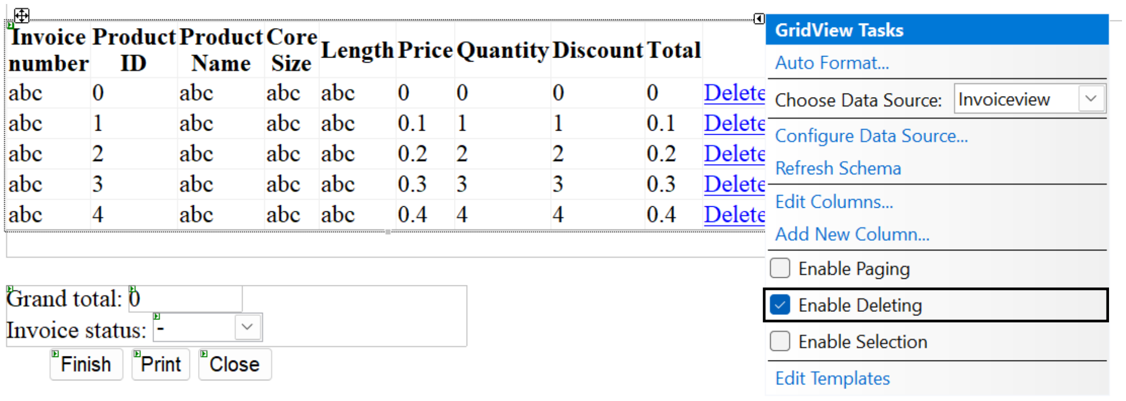
Done

Date of order	Date of due	Invoice year	Invoice number	Customer name	Salesperson name	Product name	Price	Discounts	Quantity	Total	Status
3/13/2023 12:00:00 AM	4/12/2023 12:00:00 AM	2023	8	Grace	Bima	Tan tape	12000	1	1	11880	Not paid
3/13/2023 12:00:00 AM	4/12/2023 12:00:00 AM	2023	8	Grace	Bima	Jtape	15000	0	1	15000	Not paid
3/15/2023 12:00:00 AM	4/14/2023 12:00:00 AM	2023	10	Grace	Bima	Tan tape	12000	1	1	11880	Not paid

Figure 4.2: Data binding working properly

5. Customizing grid view to enable deletion and direct calculation form its column

Delete option is required for the grid view in the invoice table page in the case that the user inputted wrong details of the invoice. Not only does this method enhance user interface because not needing to redo the wrong invoice all over again, however, employed along with the automatic calculation or sum of one column of the grid view, the user will be granted time saving functionality and lesser errors as it gets updated real-time.



Invoice number	Product ID	Product Name	Core Size	Length	Price	Quantity	Discount	Total	
abc	0	abc	abc	abc	0	0	0	0	Delete
abc	1	abc	abc	abc	0.1	1	1	0.1	Delete
abc	2	abc	abc	abc	0.2	2	2	0.2	Delete
abc	3	abc	abc	abc	0.3	3	3	0.3	Delete
abc	4	abc	abc	abc	0.4	4	4	0.4	Delete

Grand total: 0
Invoice status: -

Enable Paging ☐
Enable Deleting ✓
Enable Selection ☐

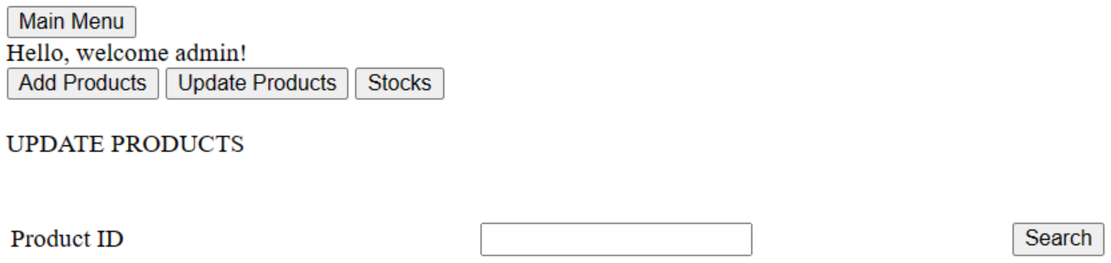
Figure 5.1: Enable deleting in grid view

```
    'CALCULATE THE TOTAL TRANSACTION DIRECTLY FROM THE GRID VIEW
    Dim total, total1 As Double
    For i As Integer = 0 To GridView1.Rows.Count - 1
        total = Convert.ToInt32(GridView1.Rows(i).Cells(8).Text)
        total1 = total1 + total
    Next
    Label4.Text = total1
End Sub
```

Figure 5.2: Code for calculating the grand total by finding the sum of the grid view's 'total' column

6. Use of panels

I made use of panels to enhance user's comfortability and improve their interface. I utilize panels to promote abstraction and authentication. Only if the present check and ID is matched with the database, the search button will make the panel containing the update form visible.

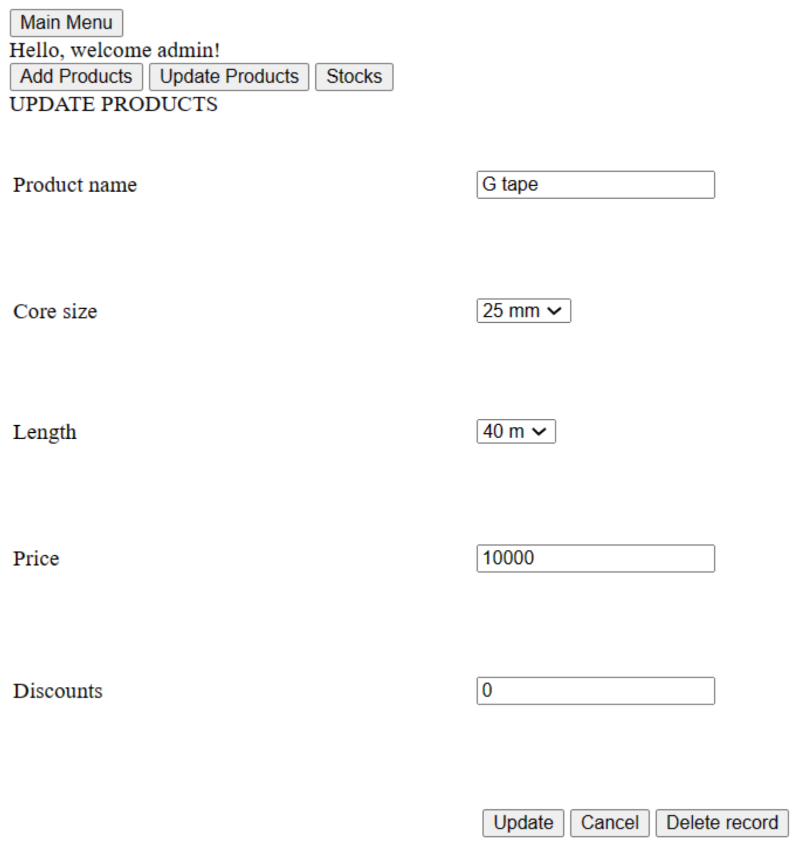


Hello, welcome admin!

UPDATE PRODUCTS

Product ID

Figure 6.1: Panel not showing



Hello, welcome admin!

UPDATE PRODUCTS

Product name	G tape
Core size	25 mm
Length	40 m
Price	10000
Discounts	0

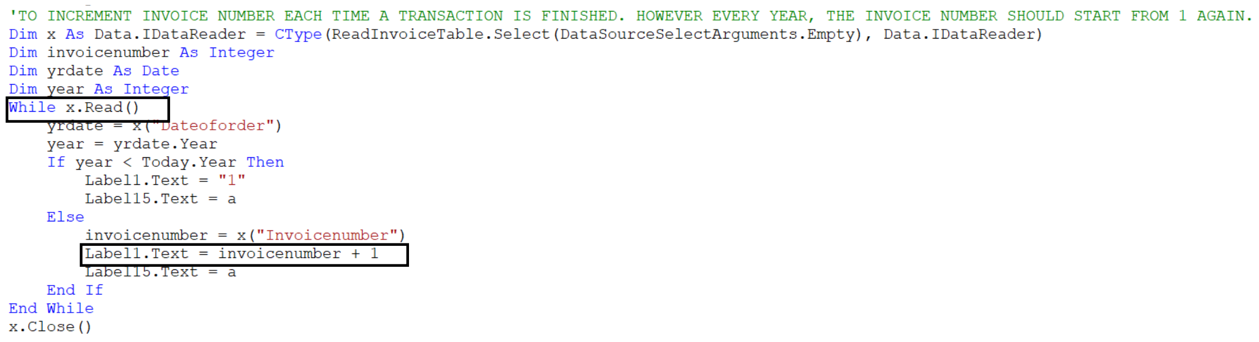
Figure 6.2: Panel containing information about the product and actions for update is visible

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim x As Data.IDataReader = CType(Readproducts.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean
    f = False
    While x.Read()
        If TextBox5.Text IsNot ("") And TextBox5.Text = x("ProductID") Then 'search button to validate product
            f = True
            Panel3.Visible = True
            Panel2.Visible = False
            TextBox6.Text = x("Productname")
            DropDownList1.Text = x("Coresize")
            DropDownList2.Text = x("Length")
            TextBox7.Text = x("Price")
            TextBox8.Text = x("Discounts")
        End If
    End While
    x.Close()
    If f = False Then
        MsgBox("Please enter a correct product ID")
    End If
End Sub
```

Figure 6.3: Code for showing and hiding the panels

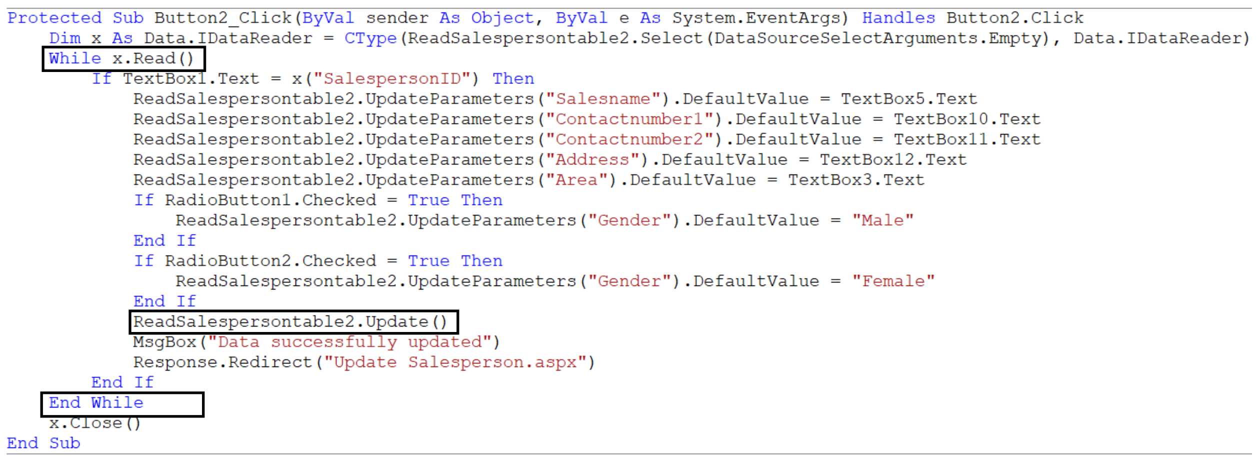
7. Use of condition-based loops to increase efficiency

I make use of condition-based loops for two different functions; to make manual auto number ID to prevent errors, and read data. I use loop to create auto numbers to prevent duplication of data which may cause errors from the primary key. I use loop to read data to execute the DML technique and perform validation.



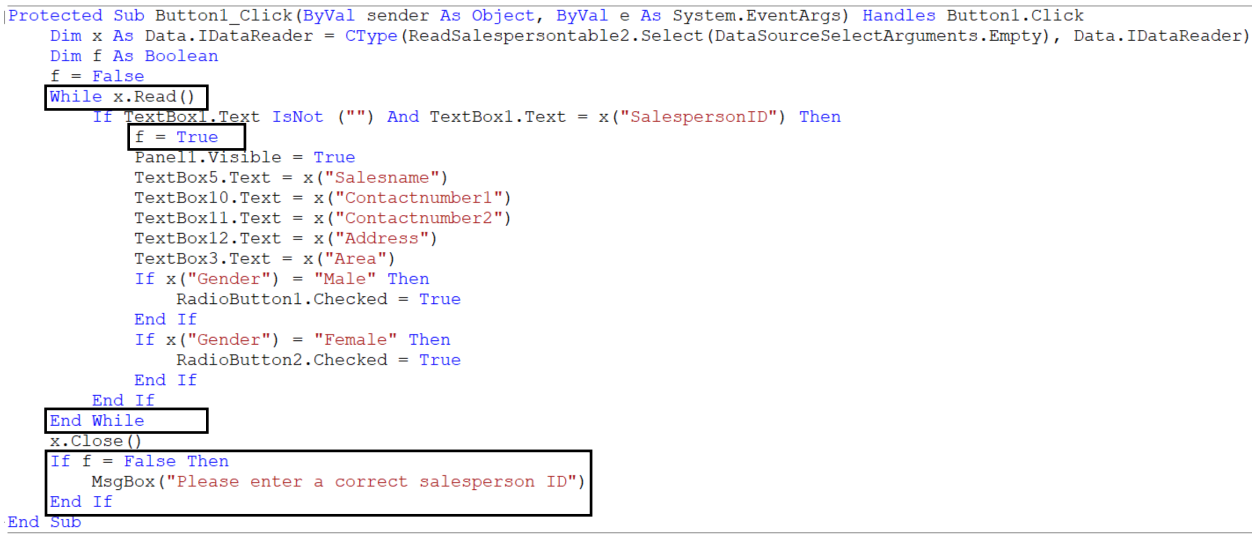
```
'TO INCREMENT INVOICE NUMBER EACH TIME A TRANSACTION IS FINISHED. HOWEVER EVERY YEAR, THE INVOICE NUMBER SHOULD START FROM 1 AGAIN.
Dim x As Data.IDataReader = CType(ReadInvoiceTable.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
Dim invoicenumber As Integer
Dim yrdate As Date
Dim year As Integer
While x.Read()
    yrdate = x("Dateoforder")
    year = yrdate.Year
    If year < Today.Year Then
        Label1.Text = "1"
        Label15.Text = a
    Else
        invoicenumber = x("Invoicenumber")
        Label1.Text = invoicenumber + 1
        Label15.Text = a
    End If
End While
x.Close()
```

Figure 7.1: While-end while loop to automatically increase invoice ID



```
Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim x As Data.IDataReader = CType(ReadSalespersontable2.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    While x.Read()
        If TextBox1.Text = x("SalespersonID") Then
            ReadSalespersontable2.UpdateParameters("Salesname").DefaultValue = TextBox5.Text
            ReadSalespersontable2.UpdateParameters("Contactnumber1").DefaultValue = TextBox10.Text
            ReadSalespersontable2.UpdateParameters("Contactnumber2").DefaultValue = TextBox11.Text
            ReadSalespersontable2.UpdateParameters("Address").DefaultValue = TextBox12.Text
            ReadSalespersontable2.UpdateParameters("Area").DefaultValue = TextBox3.Text
            If RadioButton1.Checked = True Then
                ReadSalespersontable2.UpdateParameters("Gender").DefaultValue = "Male"
            End If
            If RadioButton2.Checked = True Then
                ReadSalespersontable2.UpdateParameters("Gender").DefaultValue = "Female"
            End If
            ReadSalespersontable2.Update()
            MsgBox("Data successfully updated")
            Response.Redirect("Update Salesperson.aspx")
        End If
    End While
    x.Close()
End Sub
```

Figure 7.2: While-end while loop to read the database and update data

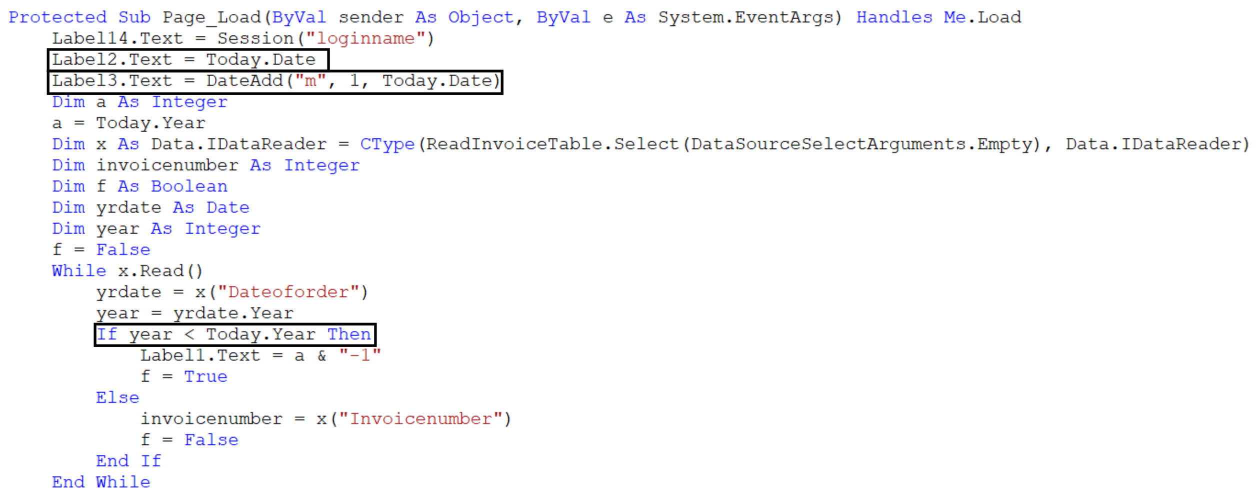


```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim x As Data.IDataReader = CType(ReadSalespersontable2.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean
    f = False
    While x.Read()
        If TextBox1.Text IsNot ("") And TextBox1.Text = x("SalespersonID") Then
            f = True
            Panel1.Visible = True
            TextBox5.Text = x("Salesname")
            TextBox10.Text = x("Contactnumber1")
            TextBox11.Text = x("Contactnumber2")
            TextBox12.Text = x("Address")
            TextBox3.Text = x("Area")
            If x("Gender") = "Male" Then
                RadioButton1.Checked = True
            End If
            If x("Gender") = "Female" Then
                RadioButton2.Checked = True
            End If
        End If
    End While
    x.Close()
    If f = False Then
        MsgBox("Please enter a correct salesperson ID")
    End If
End Sub
```

Figure 7.3: While-end while loop to read the database and compare with the user input

8. Use of date function

I made use of date function for three purposes; to display today's date, add and compare date for invoices and serves as the time period for the monthly report. The invoice due date is programmed to be 30 days after the date of order and the monthly report displays the invoices starting from the start of the month.



```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Label14.Text = Session("loginname")
    Label2.Text = Today.Date
    Label3.Text = DateAdd("m", 1, Today.Date)
    Dim a As Integer
    a = Today.Year
    Dim x As Data.IDataReader = CType(ReadInvoiceTable.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim invoicenumber As Integer
    Dim f As Boolean
    Dim yrdate As Date
    Dim year As Integer
    f = False
    While x.Read()
        yrdate = x("Dateoforder")
        year = yrdate.Year
        If year < Today.Year Then
            Label1.Text = a & "-1"
            f = True
        Else
            invoicenumber = x("Invoicenumber")
            f = False
        End If
    End While
```

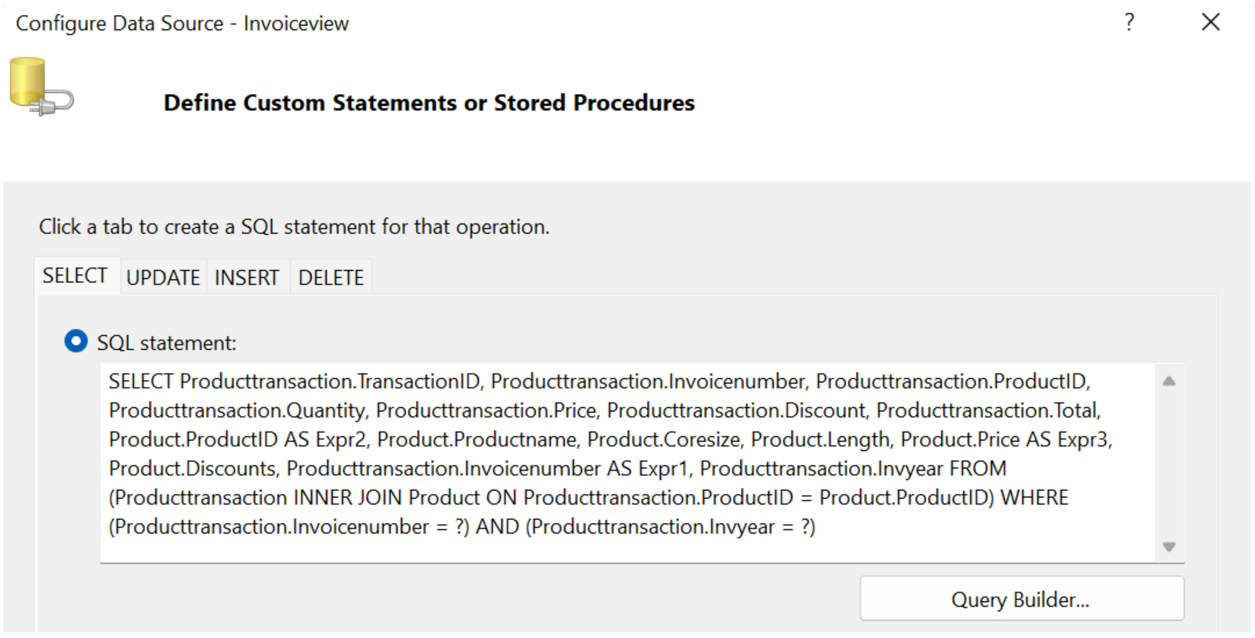
Figure 8.1: Code for date function to display today's date, add date and compare dates

```
Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click
    Panel1.Visible = False
    Panel2.Visible = True
    Panel3.Visible = False
    Dim d1, d2 As Date
    d1 = New DateTime(Today.Date.Year, Today.Date.Month, 1) 'first day of the month
    d2 = d1.AddMonths(1).AddDays(-1) 'last day of the month
    Label4.Text = d1
    Label5.Text = d2
End Sub
```

Figure 8.2: Displaying the time period for easier filter of grid view in the monthly report

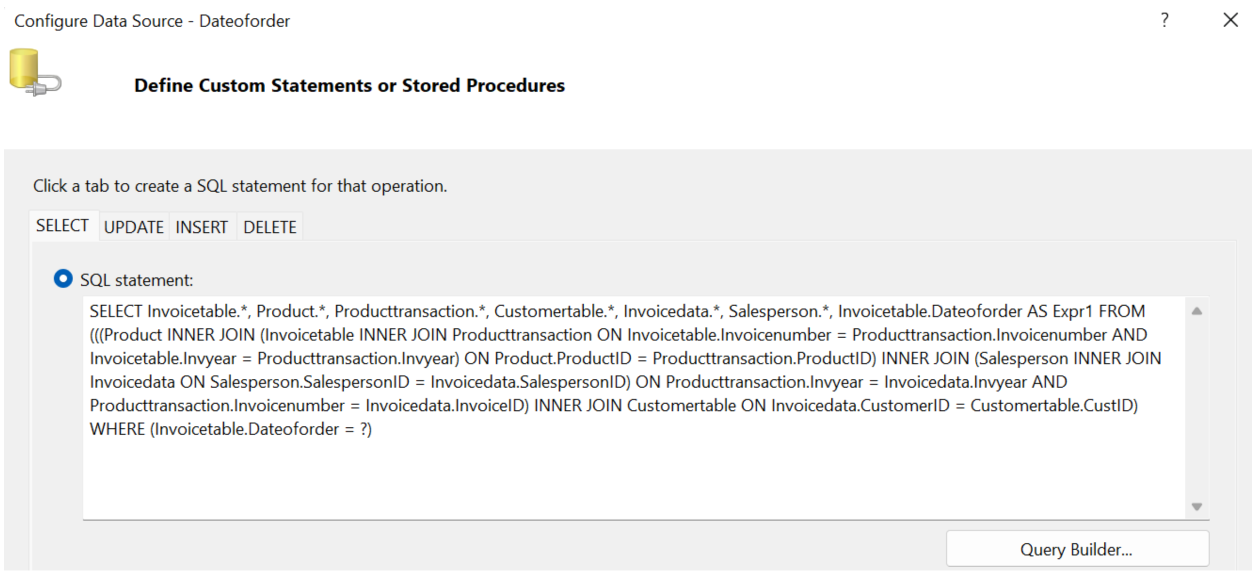
9. Use of complex queries

By using normalization and relationships, complex queries are made to be able to display data in webpages in a more systematic way which will be more easily understood by the user as well as making sure that data is filtered so user can search according to the required criteria.


Configure Data Source - Invoiceview
Define Custom Statements or Stored Procedures
Click a tab to create a SQL statement for that operation.
SELECT
UPDATE
INSERT
DELETE
SQL statement:
Query Builder...

```
SELECT Producttransaction.TransactionID, Producttransaction.Invoicenumber, Producttransaction.ProductID,
Producttransaction.Quantity, Producttransaction.Price, Producttransaction.Discount, Producttransaction.Total,
Product.ProductID AS Expr2, Product.Productname, Product.Coresize, Product.Length, Product.Price AS Expr3,
Product.Discounts, Producttransaction.Invoicenumber AS Expr1, Producttransaction.Invyear FROM
(Producttransaction INNER JOIN Product ON Producttransaction.ProductID = Product.ProductID) WHERE
(Producttransaction.Invoicenumber = ?) AND (Producttransaction.Invyear = ?)
```

Figure 9.1: Complex queries to display invoice table according to the invoice number and invoice year


Configure Data Source - Dateoforder

Define Custom Statements or Stored Procedures

Click a tab to create a SQL statement for that operation.

SELECT | UPDATE | INSERT | DELETE

(●) SQL statement:

```
SELECT Invoicetable.*, Product.*, Producttransaction.*, Customertable.*, Invoicedata.*, Salesperson.*, Invoicetable.Dateoforder AS Expr1 FROM
(((Product INNER JOIN (Invoicetable INNER JOIN Producttransaction ON Invoicetable.Invoicenumber = Producttransaction.Invoicenumber AND
Invoicetable.Invyear = Producttransaction.Invyear) ON Product.ProductID = Producttransaction.ProductID) INNER JOIN (Salesperson INNER JOIN
Invoicedata ON Salesperson.SalespersonID = Invoicedata.SalespersonID) ON Producttransaction.Invyear = Invoicedata.Invyear AND
Producttransaction.Invoicenumber = Invoicedata.InvoiceID) INNER JOIN Customertable ON Invoicedata.CustomerID = Customertable.CustID)
WHERE (Invoicetable.Dateoforder = ?)
```

Query Builder...

Figure 9.2: Complex queries to search invoices based on their date of order

10. Use of normalization and relationships

Normalization is used in the database in order to avoid redundancy of data and to save storage space. Relationships are also created to ensure that complex queries can be made.

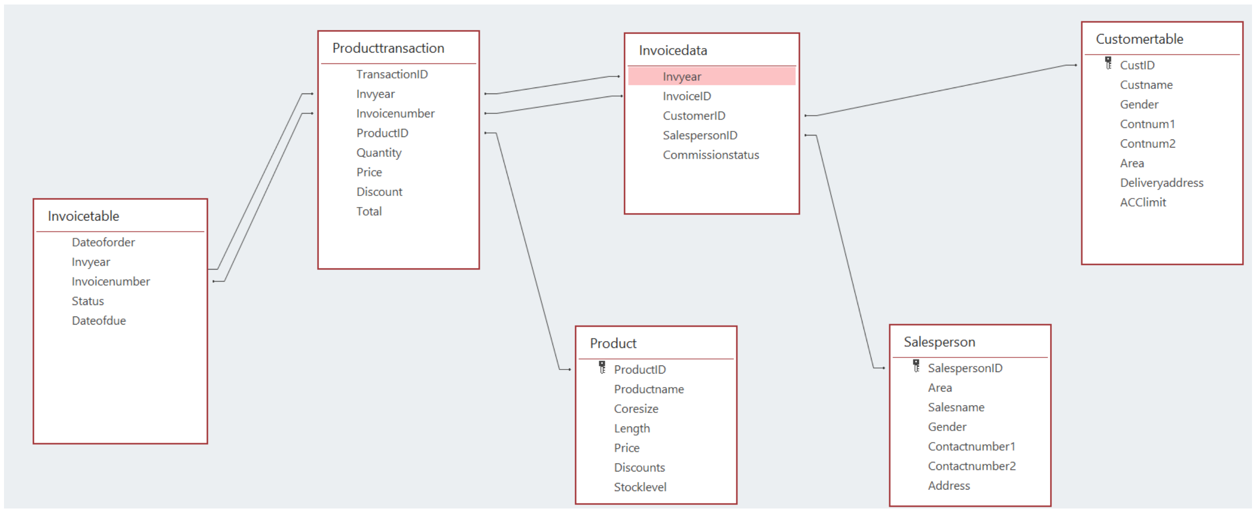


Figure 10.1: Relationship used in the database

11. Data manipulation language (select, insert, update, delete)

I made use of DML to perform changes in the data of the database such as insert, update, deleting of products or customers or salesperson's details.

```
Protected Sub Button3_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button3.Click
    SetSalespersontable.InsertParameters("SalespersonID").DefaultValue = TextBox4.Text
    SetSalespersontable.InsertParameters("Salesname").DefaultValue = TextBox6.Text
    SetSalespersontable.InsertParameters("Contactnumber1").DefaultValue = TextBox7.Text
    SetSalespersontable.InsertParameters("Contactnumber2").DefaultValue = TextBox8.Text
    SetSalespersontable.InsertParameters("Address").DefaultValue = TextBox9.Text
    SetSalespersontable.InsertParameters("Area").DefaultValue = TextBox2.Text
    If RadioButton3.Checked = True Then
        SetSalespersontable.InsertParameters("Gender").DefaultValue = "Male"
    Else
        SetSalespersontable.InsertParameters("Gender").DefaultValue = "Female"
    End If
    SetSalespersontable.Insert()
    MsgBox("Data successfully saved")
    Response.Redirect("Update Salesperson.aspx")
End Sub
```

```
<asp:AccessDataSource ID="SetSalespersontable" runat="server" DataFile="~/CS database.mdb"
    SelectCommand="SELECT DISTINCT * FROM [Salesperson] "
    InsertCommand = "insert into Salesperson (SalespersonID, Salesname, Gender, Contactnumber1, Contactnumber2, Address, Area)
    values (@SalespersonID, @Salesname, @Gender, @Contactnumber1, @Contactnumber2, @Address, @Area) " >
     <InsertParameters>
               <asp:Parameter Name= "SalespersonID" DefaultValue = "" />
               <asp:Parameter Name = "Salesname" DefaultValue ="" />
               <asp:Parameter Name = "Gender" DefaultValue ="" />
               <asp:Parameter Name= "Contactnumber1" DefaultValue = "" />
               <asp:Parameter Name = "Contactnumber2" DefaultValue ="" />
               <asp:Parameter Name = "Address" DefaultValue ="" />
               <asp:Parameter Name = "Area" DefaultValue ="" />
               </InsertParameters>
    </asp:AccessDataSource>
```

Figure 11.1: Select and Insert code and parameters

```
While x.Read()
    If TextBox1.Text = x("SalespersonID") Then
        ReadSalespersontable2.UpdateParameters("Salesname").DefaultValue = TextBox5.Text
        ReadSalespersontable2.UpdateParameters("Contactnumber1").DefaultValue = TextBox10.Text
        ReadSalespersontable2.UpdateParameters("Contactnumber2").DefaultValue = TextBox11.Text
        ReadSalespersontable2.UpdateParameters("Address").DefaultValue = TextBox12.Text
        ReadSalespersontable2.UpdateParameters("Area").DefaultValue = TextBox3.Text
        If RadioButton1.Checked = True Then
            ReadSalespersontable2.UpdateParameters("Gender").DefaultValue = "Male"
        End If
        If RadioButton2.Checked = True Then
            ReadSalespersontable2.UpdateParameters("Gender").DefaultValue = "Female"
        End If
        ReadSalespersontable2.Update()
        MsgBox("Data successfully updated")
        Response.Redirect("Update Salesperson.aspx")
    End If
End While
```

```
    Protected Sub Button8_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button8.Click
        MsgBox("Delete record?", MsgBoxStyle.YesNo)
        If MsgBoxResult.Yes Then
            ReadSalespersontable2.DeleteParameters("SalespersonID").DefaultValue = TextBox1.Text
            ReadSalespersontable2.Delete()
            MsgBox("Record successfully deleted")
            Response.Redirect("Update salesperson.aspx")
        End If
    End Sub
End Class
```

```
<asp:AccessDataSource ID="ReadSalespersontable2" runat="server" DataFile="~/CS database.mdb"
    SelectCommand="SELECT * FROM [Salesperson] ORDER BY [SalespersonID]" DataSourceMode="DataReader"
     DeleteCommand="DELETE FROM [Salesperson] WHERE [SalespersonID] = @SalespersonID"
     UpdateCommand="update Salesperson
                   set Area=@Area, Salesname=@Salesname, Gender=@Gender, Contactnumber1=@Contactnumber1,
                   Contactnumber2=@Contactnumber2, Address=@Address
                   where SalespersonID=@SalespersonID">
    <UpdateParameters>
    <asp:Parameter Name="Area" DefaultValue=""/>
    <asp:Parameter Name="Salesname" DefaultValue=""/>
    <asp:Parameter Name="Gender" DefaultValue=""/>
    <asp:Parameter Name="Contactnumber1" DefaultValue=""/>
    <asp:Parameter Name="Contactnumber2" DefaultValue=""/>
    <asp:Parameter Name="Address" DefaultValue=""/>
    <asp:ControlParameter ControlID="Textbox1" PropertyName="Text"  Name="SalespersonID" DefaultValue=""/>
    </UpdateParameters>
    <DeleteParameters>
<asp:Parameter Name="SalespersonID" Type="Int32" />
</DeleteParameters>
    </asp:AccessDataSource>
```

Figure 11.2: Select, delete and update code and parameters

12. Use of object to read database

Object means a variable is declared as a data reader. This is done in order to be able to read and find the specific row of data in the database. It can also be utilized to compare the data in the database with the user input and take further actions.

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    'To check whether inputted username and password are correct or matching with the database
    Dim x As Data.IDataReader = CType(Readlogin.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean = False
    While x.Read()
        If TextBox1.Text = x("Uname") And TextBox2.Text = x("Pword") Then
            Session("loginname") = x("Uname") 'session to store user's username to enhance user interface
            f = True
        End If
    End While
    If f = True Then
        MsgBox("You have logged in successfully")
        Response.Redirect("Main menu.aspx")
    Else
        MsgBox("Incorrect username or password")
    End If
End Sub
```

Figure 12.1: Code to read and compare username and password

```
Protected Sub Button7_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button7.Click
    Dim x As Data.IDataReader = CType(ReadCustomertable.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    Dim f As Boolean
    f = False
    While x.Read()
        If TextBox3.Text IsNot ("") And TextBox3.Text = x("CustID") Then
            f = True
            Panel3.Visible = True
            TextBox4.Text = x("Custname")
            TextBox5.Text = x("Contnum1")
            TextBox6.Text = x("Contnum2")
            DropDownList5.Text = x("Area")
            TextBox10.Text = x("Deliveryaddress")
            TextBox12.Text = x("ACClimit")
            If x("Gender") = "Male" Then
                RadioButton3.Checked = True
            End If
            If x("Gender") = "Female" Then
                RadioButton4.Checked = True
            End If
        End If
    End While
    If f = False Then
        MsgBox("Please enter a correct customer ID")
    End If
End Sub
```

Figure 12.2: Code to read and display customers data in the update form

13. Nested loop

I use nested loops to enable reading multiple tables using multiple readers simultaneously, perform sorting in two-dimensional array and to display output in list boxes.

```
Dim b As Data.IDataReader = CType(Readproducttransaction.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
While b.Read()
    If Label1.Text = b("Invoicenumber") And Label15.Text = b("Invyear") Then
        q1 = b("Quantity")
        prod = b("ProductID")
        Dim c As Data.IDataReader = CType(Readproducts.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
        While c.Read()
            If prod = c("ProductID") Then
                q2 = c("Stocklevel")
                q2 = q2 - q1 'minus the demand from the stock level
            End If
        End While
        Readproducts.UpdateParameters("Stocklevel").DefaultValue = q2
        Readproducts.Update()
        c.Close()
    End If
End While
b.Close()
```

Figure 13.1: Nested while loop used in reading tables using multiple readers simultaneously

```
Dim i, j, temp As Integer
For i = 0 To count - 2
    For j = i To count - 1
        If product(i, 1) < product(j, 1) Then
            temp = product(j, 1)
            product(j, 1) = product(i, 1)
            product(i, 1) = temp
            temp = product(j, 0)
            product(j, 0) = product(i, 0)
            product(i, 0) = temp
        End If
    Next
Next
```

Figure 13.2: Nested for loop to sort items in two-dimensional array

```
For i = 0 To 4
    ListBox2.Items.Add(revenue(i)) 'adding to list boxes
    Dim o As Data.IDataReader = CType(ReadsalesID.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    While o.Read()
        If salesid(i) = o("SalespersonID") Then
            ListBox1.Items.Add(o("Salesname"))
        End If
    End While
    o.Close()
Next
```

Figure 13.3: Nested for and while loop to display the sorted items in the list boxes

14. Bubble Sorting

I perform sorting using one-dimensional arrays and two-dimensional arrays to fulfill the client’s request and make easy to track the performance of their salespersons and products.

```
For i = 0 To count - 2
    For j = i To count - 1
        If revenue(i) < revenue(j) Then
            temp = revenue(j)
            revenue(j) = revenue(i)
            revenue(i) = temp
            temp = salesid(j)
            salesid(j) = salesid(i)
            salesid(i) = temp
        End If
    Next
Next
```

Figure 14.1: Sorting using one-dimensional array

```
Dim i, j, temp As Integer
For i = 0 To count - 2
    For j = i To count - 1
        If product(i, 1) < product(j, 1) Then
            temp = product(j, 1)
            product(j, 1) = product(i, 1)
            product(i, 1) = temp
            temp = product(j, 0)
            product(j, 0) = product(i, 0)
            product(i, 0) = temp
        End If
    Next
Next
```

Figure 14.2: Sorting using two-dimensional array

15. Two-dimensional array

Two-dimensional array is utilized to store the list of products (row) as well as their sales revenue (column) which not only reduces the storage capacity required but also achieve fewer lines of code with less complex coding in the sorting technique.

```
For i = 0 To 4
    ListBox4.Items.Add(product(i, 1))
    Dim o As Data.IDataReader = CType(Readproductid.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    While o.Read()
        If product(i, 0) = o("ProductID") Then
            ListBox3.Items.Add(o("Productname"))
        End If
    End While
Next
```

Figure 15.1: Nested for-while loop to display items in two-dimensional array on a list box

16. Use of list boxes

List boxes are utilized to display the output of the sorting of top 5 salespersons based on their sales revenue thus promotes abstraction and enhance user interface to track the performance of the salespersons.

```
ListBox2.Items.Clear()
ListBox1.Items.Clear()
For i = 0 To 4
    ListBox2.Items.Add(revenue(i))

    Dim o As Data.IDataReader = CType(ReadsalesID.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    While o.Read()
        If salesid(i) = o("SalespersonID") Then
            ListBox1.Items.Add(o("Salesname"))
        End If
    End While
Next
```

Figure 16.1: Displaying items in one-dimensional array on a list box

17. Use of for loops

For loops are utilized only when the programmer knows the exact number of times they want the loop to go on. In the case of sorting the top 5 salesperson or products, the array index should go from 0 to 4. Thus, by using for loops, execution time will be reduced as useless actions such as reading more indexes will be avoided.

```
ListBox2.Items.Clear()
ListBox1.Items.Clear()
For i = 0 To 4
    ListBox2.Items.Add(revenue(i))

    Dim o As Data.IDataReader = CType(ReadsalesID.Select(DataSourceSelectArguments.Empty), Data.IDataReader)
    While o.Read()
        If salesid(i) = o("SalespersonID") Then
            ListBox1.Items.Add(o("Salesname"))
        End If
    End While
Next
```

Figure 17.1: For loop to display top 5 salesperson on a list box

Total WC: 912