TECHNICAL DOCUMENTATION

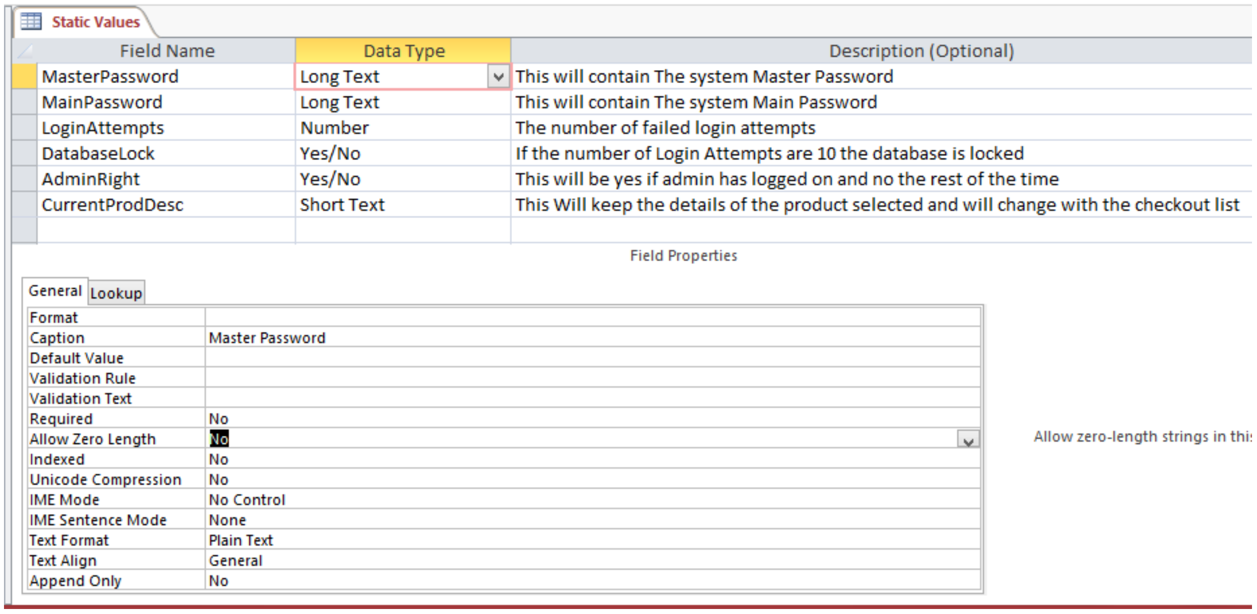
The Spaza inventory management and point of sale system was made using Microsoft Access

ACCESS DATABASE TABLES

These are the design view representations of the individual tables used in Microsoft Access.

STATIC VALUES TABLE

This is the table that stores all values that either need to remain the same when the program is closed or aren't relevant for all the other tables.



Static Values

Field Name	Data Type	Description (Optional)
MasterPassword	Long Text	This will contain The system Master Password
MainPassword	Long Text	This will contain The system Main Password
LoginAttempts	Number	The number of failed login attempts
DatabaseLock	Yes/No	If the number of Login Attempts are 10 the database is locked
AdminRight	Yes/No	This will be yes if admin has logged on and no the rest of the time
CurrentProdDesc	Short Text	This Will keep the details of the product selected and will change with the checkout list

Field Properties

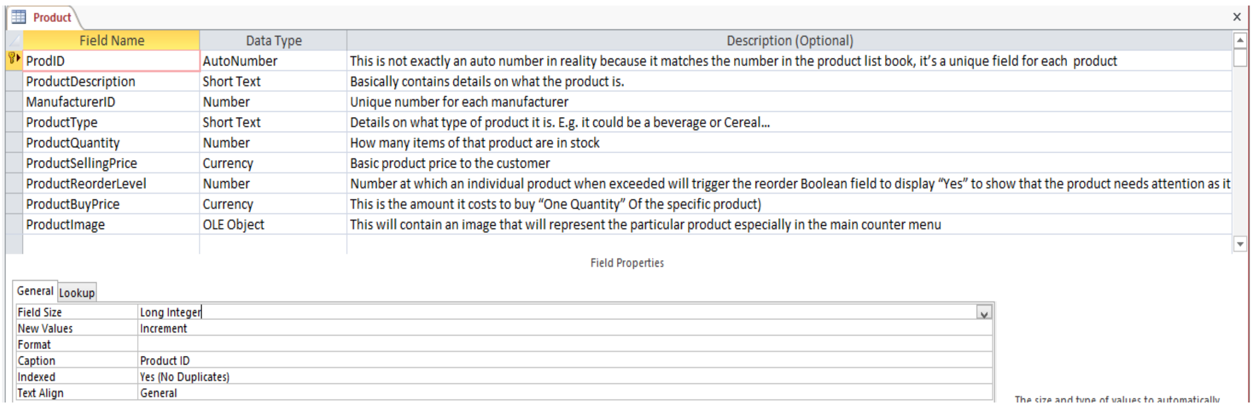
General | Lookup

Format	
Caption	Master Password
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	No
Unicode Compression	No
IME Mode	No Control
IME Sentence Mode	None
Text Format	Plain Text
Text Align	General
Append Only	No

Allow zero-length strings in thi

PRODUCT TABLE

This is the table that stores all the details about the products sold by the tuck-shop.



Product

Field Name	Data Type	Description (Optional)
ProdID	AutoNumber	This is not exactly an auto number in reality because it matches the number in the product list book, it's a unique field for each product
ProductDescription	Short Text	Basically contains details on what the product is.
ManufacturerID	Number	Unique number for each manufacturer
ProductType	Short Text	Details on what type of product it is. E.g. it could be a beverage or Cereal…
ProductQuantity	Number	How many items of that product are in stock
ProductSellingPrice	Currency	Basic product price to the customer
ProductReorderLevel	Number	Number at which an individual product when exceeded will trigger the reorder Boolean field to display "Yes" to show that the product needs attention as it
ProductBuyPrice	Currency	This is the amount it costs to buy "One Quantity" Of the specific product)
ProductImage	OLE Object	This will contain an image that will represent the particular product especially in the main counter menu

Field Properties

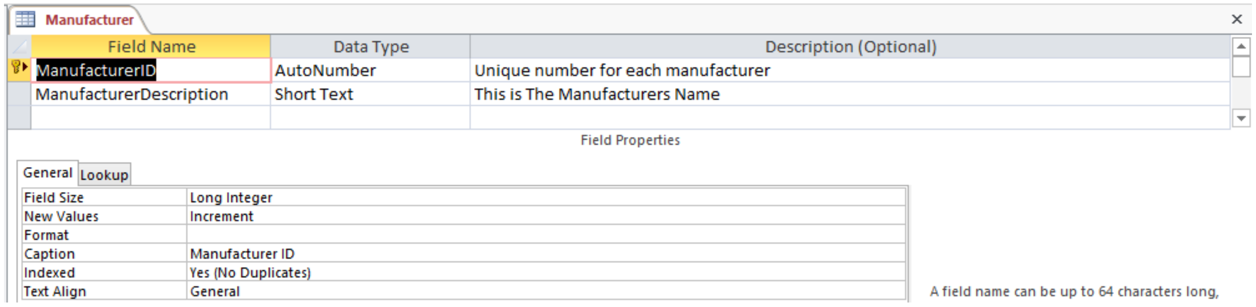
General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	Product ID
Indexed	Yes (No Duplicates)
Text Align	General

The size and type of values to automatically

MANUFACTURER TABLE

This is the table that stores the names of the individual manufacturers and their id which is used as reference in other tables.



Manufacturer

Field Name	Data Type	Description (Optional)
ManufacturerID	AutoNumber	Unique number for each manufacturer
ManufacturerDescription	Short Text	This is The Manufacturers Name

Field Properties

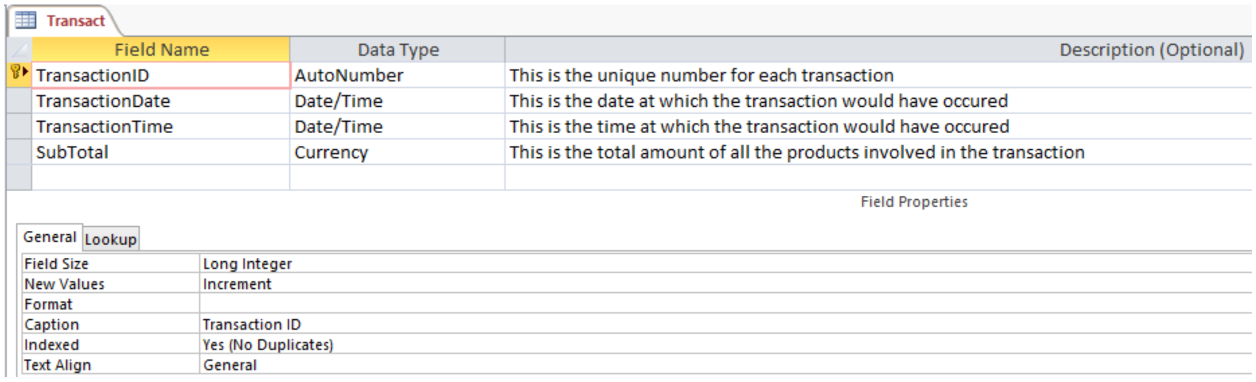
General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	Manufacturer ID
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long,

TRANSACT TABLE

This is the table that store all the details on each and every transaction.



Transact

Field Name	Data Type	Description (Optional)
TransactionID	AutoNumber	This is the unique number for each transaction
TransactionDate	Date/Time	This is the date at which the transaction would have occured
TransactionTime	Date/Time	This is the time at which the transaction would have occured
SubTotal	Currency	This is the total amount of all the products involved in the transaction

Field Properties

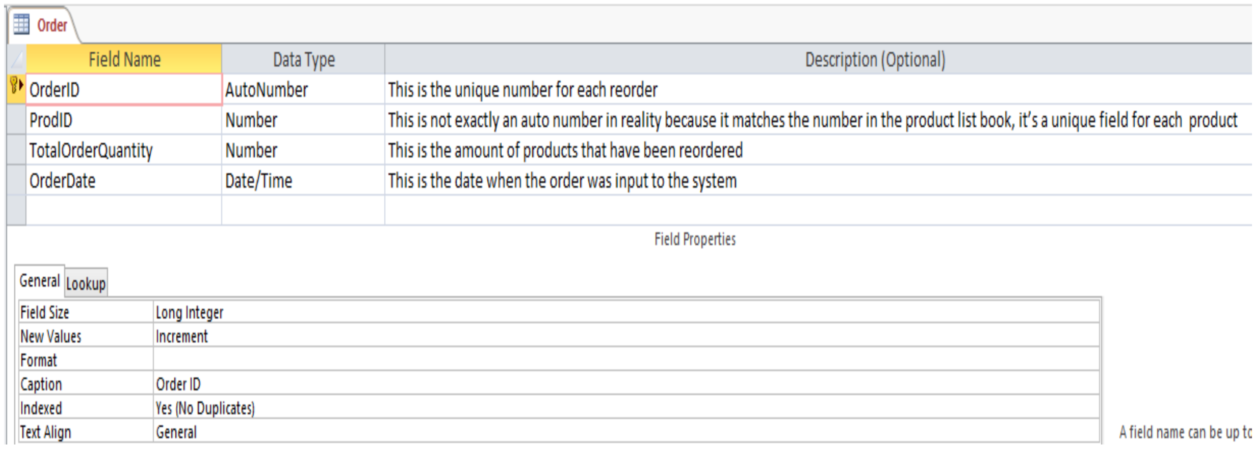
General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	Transaction ID
Indexed	Yes (No Duplicates)
Text Align	General

ORDER TABLE

Since the orders are taken from inconsistent suppliers and are recorded one product there was no use of having another table for multiple orders or suppliers.

This table however stores the details on the particular orders when they are input into the system.



Order

Field Name	Data Type	Description (Optional)
OrderID	AutoNumber	This is the unique number for each reorder
ProdID	Number	This is not exactly an auto number in reality because it matches the number in the product list book, it's a unique field for each product
TotalOrderQuantity	Number	This is the amount of products that have been reordered
OrderDate	Date/Time	This is the date when the order was input to the system

Field Properties

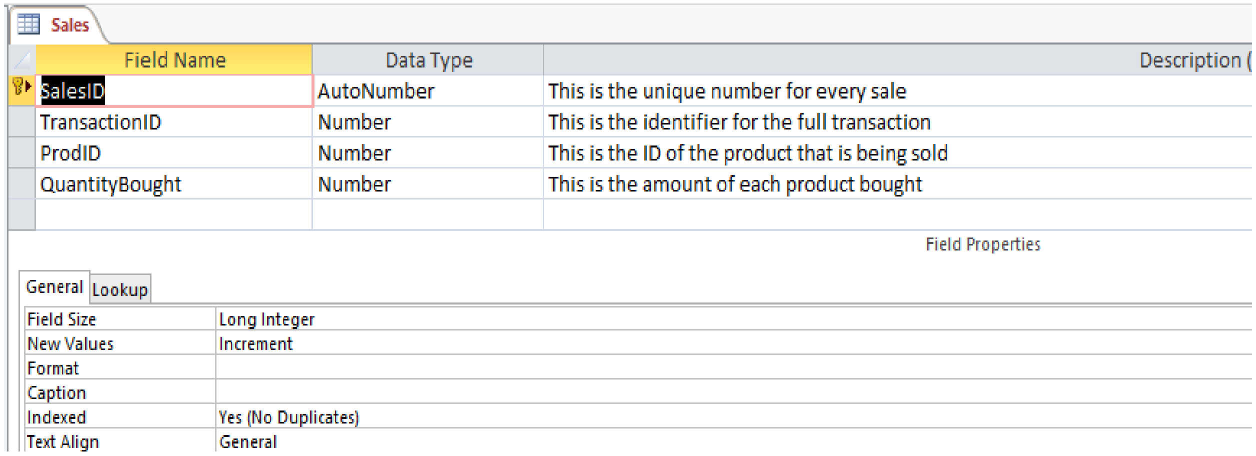
General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	Order ID
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up tc

SALES TABLE

This table contains the sales details for each product of a particular transaction as a transaction is made up of one or many product sales.



Sales

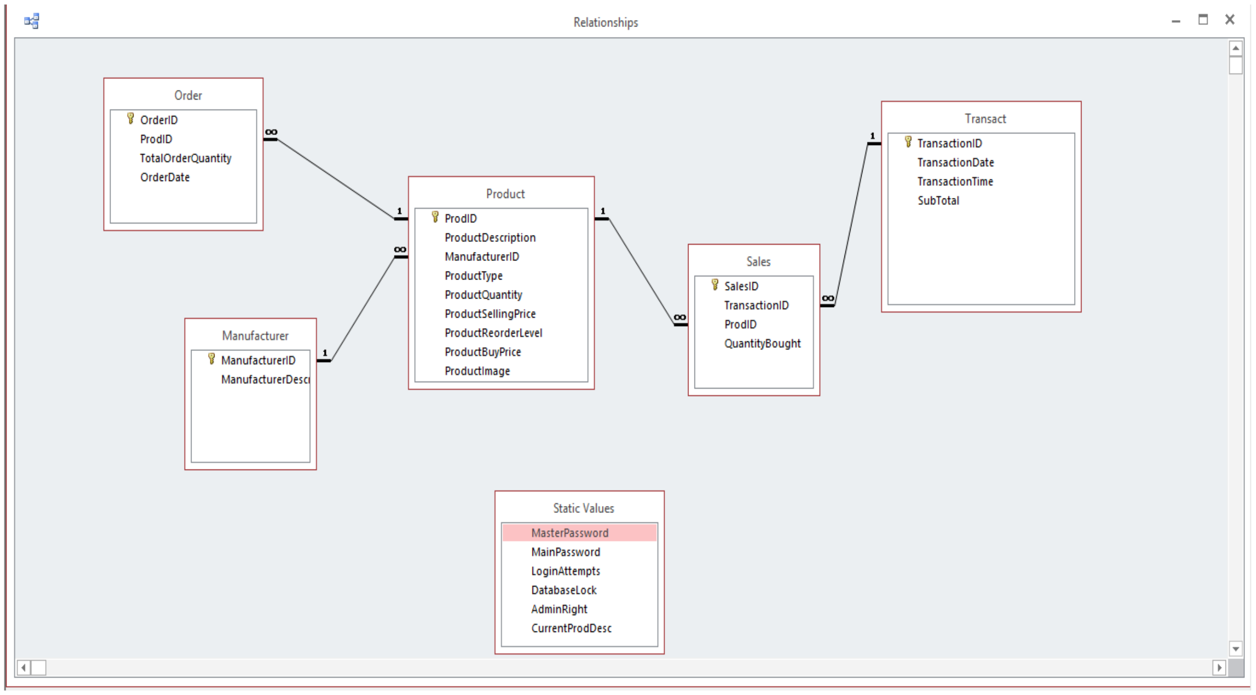
Field Name	Data Type	Description (
SalesID	AutoNumber	This is the unique number for every sale
TransactionID	Number	This is the identifier for the full transaction
ProdID	Number	This is the ID of the product that is being sold
QuantityBought	Number	This is the amount of each product bought

Field Properties

General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

ACCESS RELATIONSHIP VIEW



SPECIAL VALIDATIONS

These are the validations that were placed in the tables, the other validation rules were in the form text boxes to filter out inappropriate data.

- All ID's were made Auto-numbers because they don't need a special form. Auto-numbers naturally increment the value of the previous data entry ID and this provides a presence check of its own.
- All Dates and times have an appropriate input mask and force the correct date in form as the date is automatically set.
- The rest of the fields have a natural presence and format check in the forms

EVIDENCE

- In the table representations above all ID's have their format set to Auto-number!
- Every form where a date is recorded there is code shown in the VBA events
- Presence checks use the "is null" function as demonstrated in the login code as well as the other code or the check will be the standard validation

FORM CODE AND DESIGN STRUCTURES

These show all the programming and the design view of the form with an explanation of what exactly goes on.

For all normal views of forms reference to testing, user documentation or the system itself.

The forms are divided into categories depending on their functionalities.

GLOSSARY

Switch Button: These are buttons that simple close the current form and open the form suggested by the name. In the case that it's a back button it simply closes itself and opens the form that was previously open (except in the case of the reorder that goes back to the main menu for permission reasons as the form can be accessed from two places).

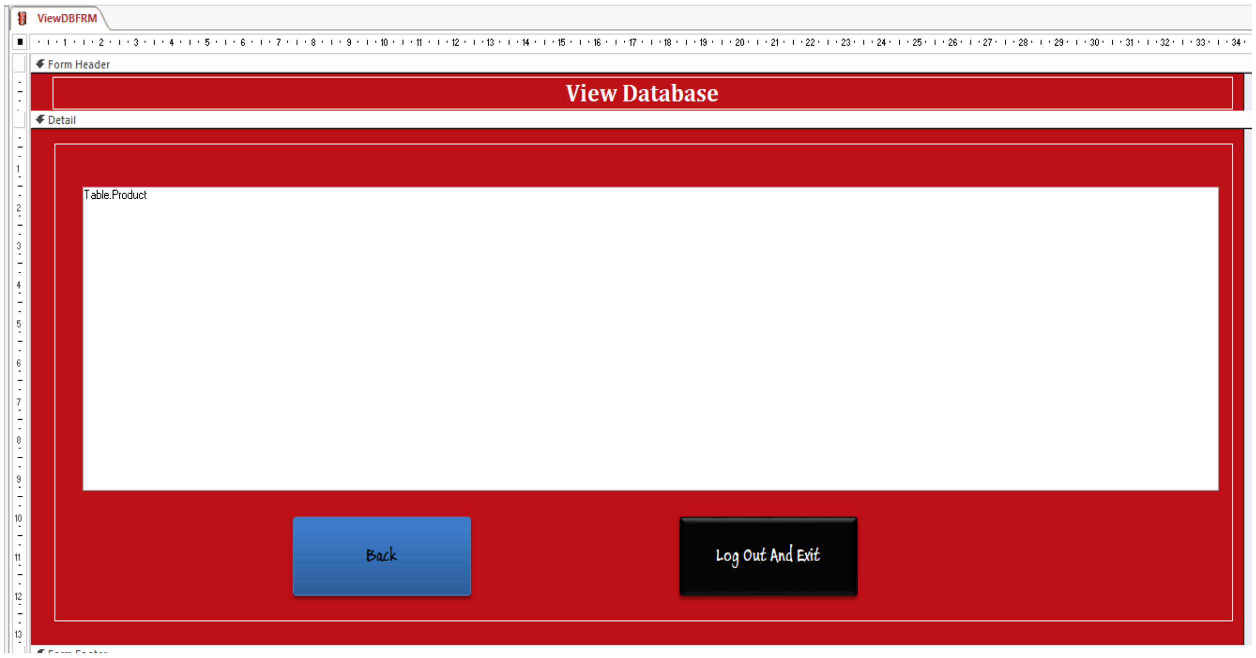
Exit Button: The system works in a way that one form is open at a time with the allowance of a pop up reorder form after login. The exit buttons close the form period.

DATA VIEW FORMS

These are the forms that simple hold the purpose of viewing data and nothing else. They either contain code to open a report, a text box with code in it to display calculated values from queries and tables.

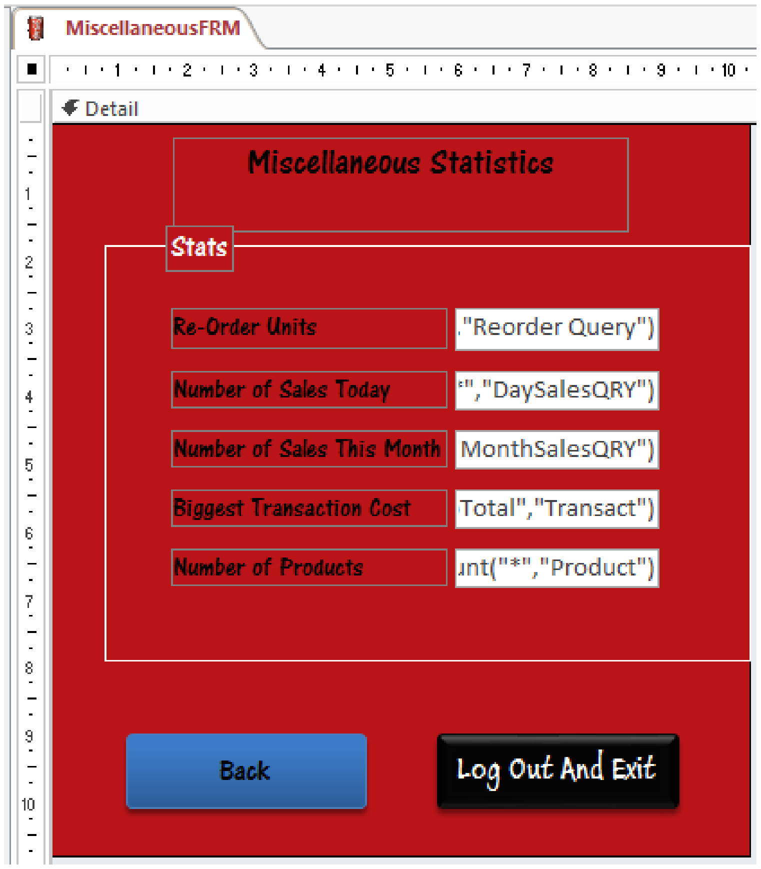
VIEW DATABASE FORM

This is a simple form with the product table at view with a sort of sub form style.



MISCELLANEOUS STATISTICS

These are calculated stats from the selected tables and queries.



Re-Order Units	,"Reorder Query")
Number of Sales Today	*","DaySalesQRY")
Number of Sales This Month	MonthSalesQRY")
Biggest Transaction Cost	Total","Transact")
Number of Products	unt("*","Product")

CODING EXPLANATION

Re order Units (=DCount("*","Reorder Query")): This counts the amount of records in the reorder query

Number of sales today (=DCount("*","DaySalesQRY")): simply counts the sales today data entries.

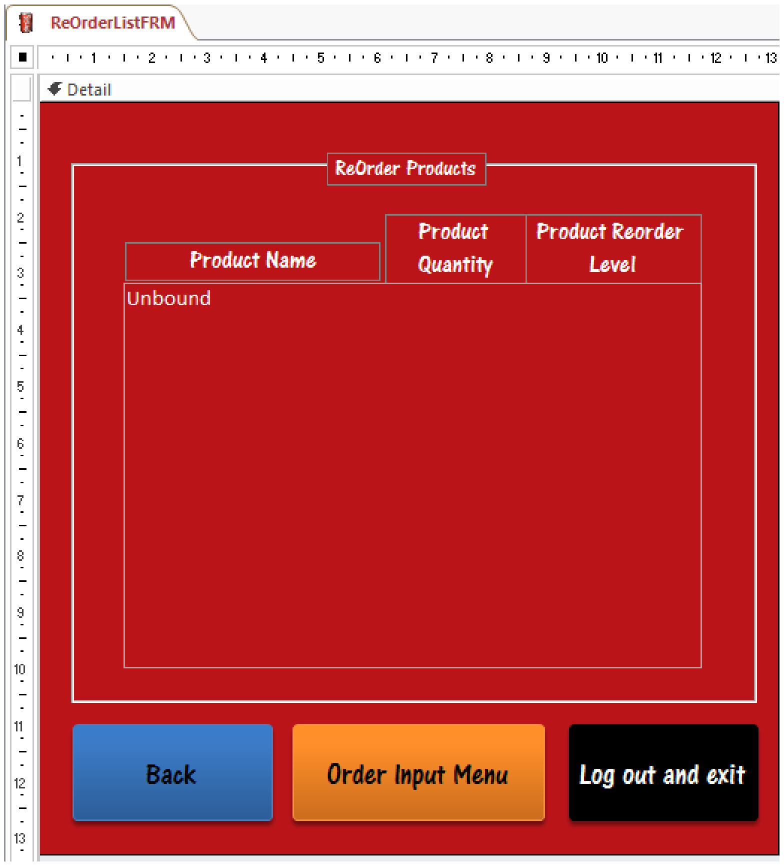
Number of sales this month (=DCount("*","MonthSalesQRY")):

Biggest transaction (=DMax("SubTotal","Transact")): the maximum transaction sub total

Number of products (): Counts the number of records in the product table

REORDER PRODUCTS FORM (LIST)

This is a basic list derived from the reorder query it just displays the products that need to be reordered in a list fashion.



The list seems unbound but in the properties under data there is code to select the selected columns (fields) from the query

CODE

SELECT [Reorder Query].[ProductDescription], [Reorder Query].[ProductQuantity], [Reorder Query].[ProductReorderLevel] FROM [Reorder Query];

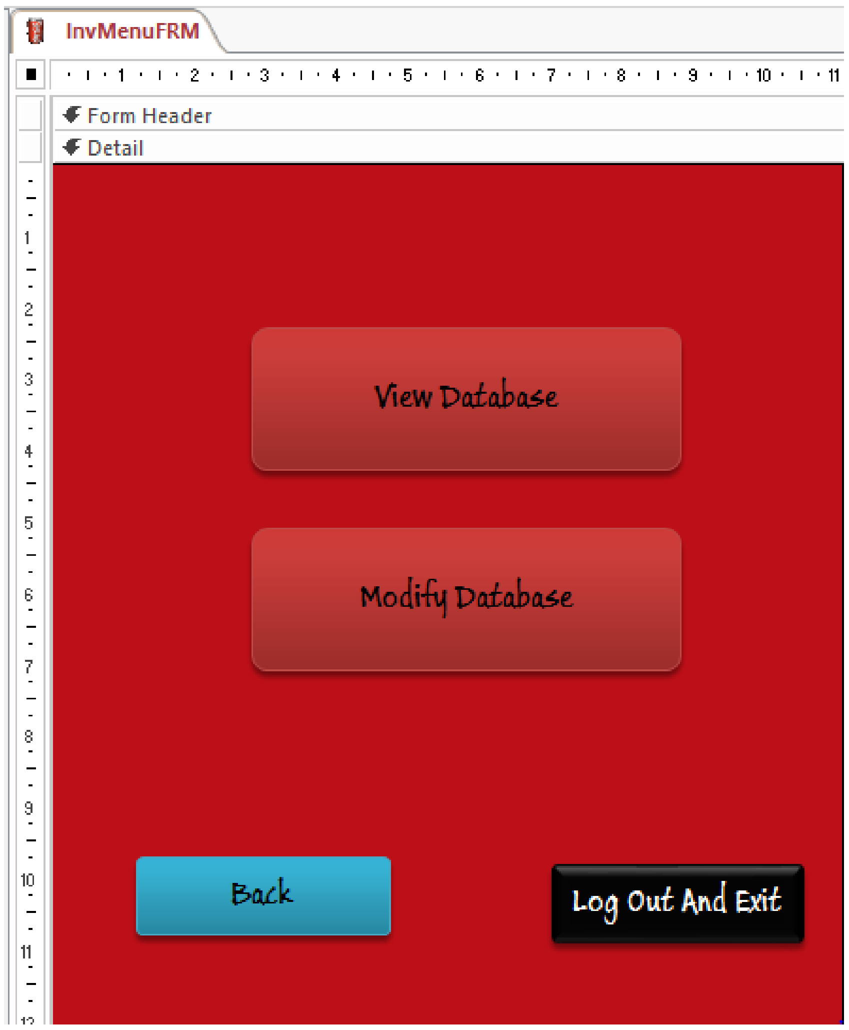
SWITCHBOARD FORMS

These are the menu forms, these forms don't have any complications in terms of code, they simple close themselves and open the form requested.

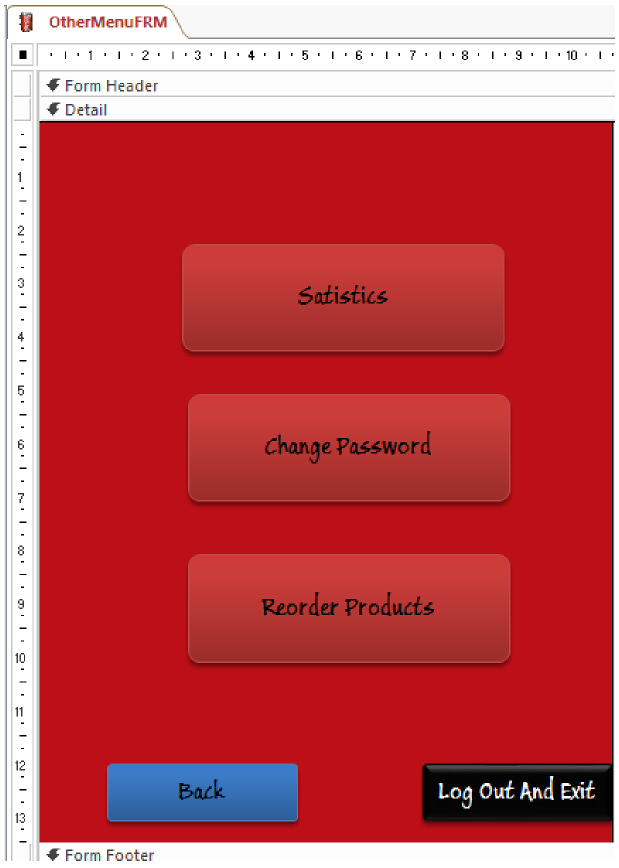
They use three commands.

- Either docmd.Close: To close the current form. Or docmd.quit to close the application
- docmd.openform: To open a Specified form.
- docmd.openreport: To open a specified report.

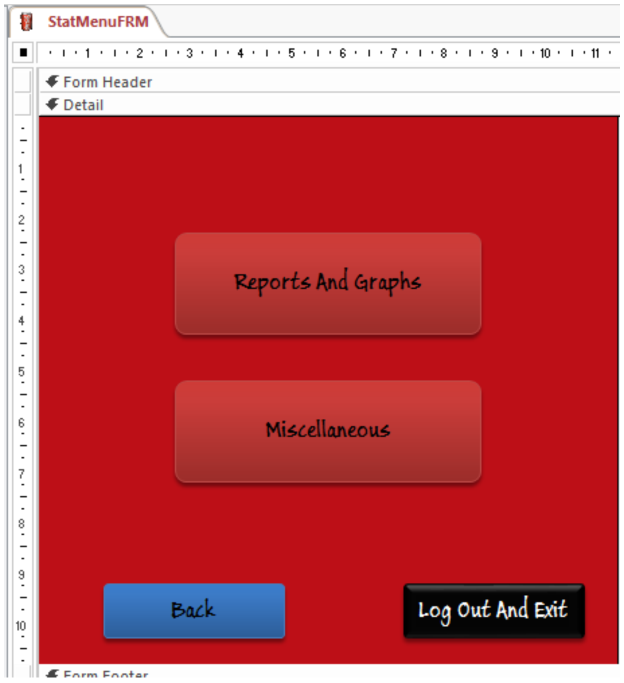
INVENTORY MENU FORM



OTHER OPTIONS FORM

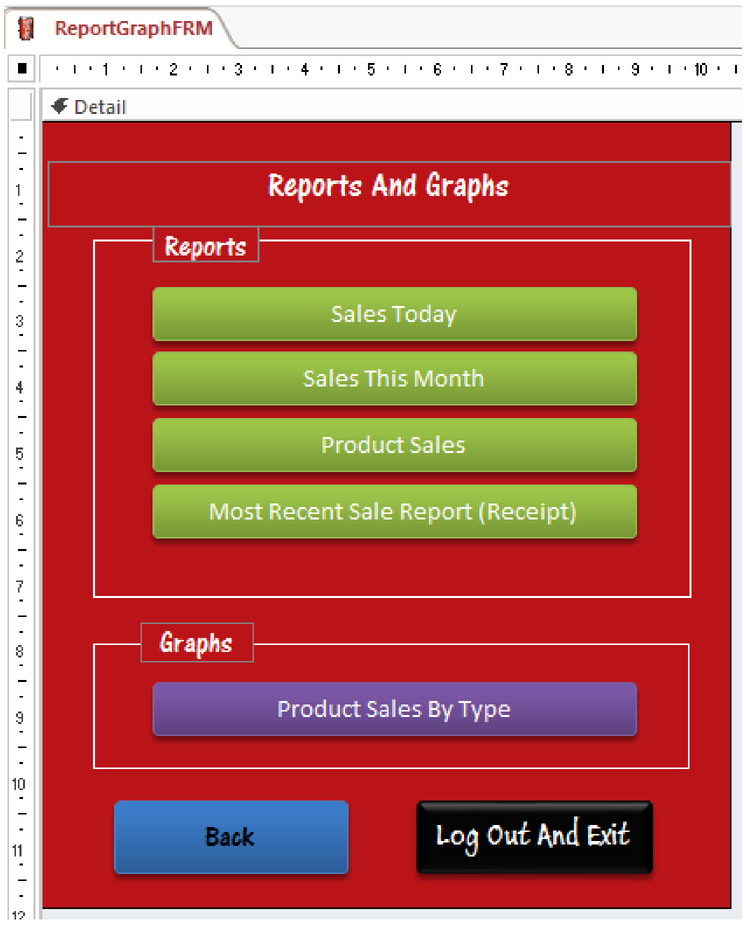


STATISTICS FORM



REPORTS AND GRAPHS FORM

Instead of then opening another form these buttons open their respective reports and graphs.

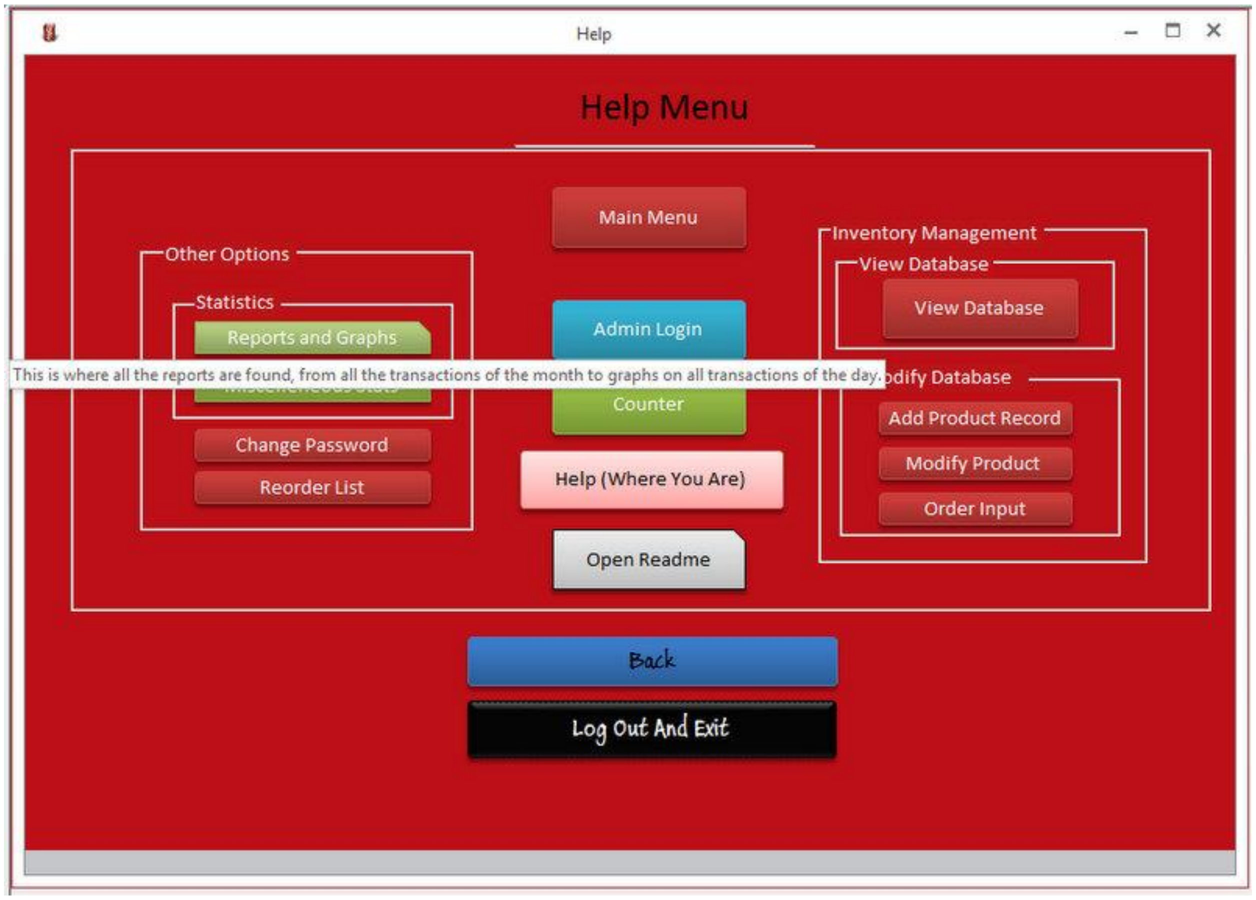


SPECIAL FORMS

These fit into one of the categories but have special features that make them more complicated.

HELP FORM

This is the form where the user accesses all the other forms ad sees what they do.



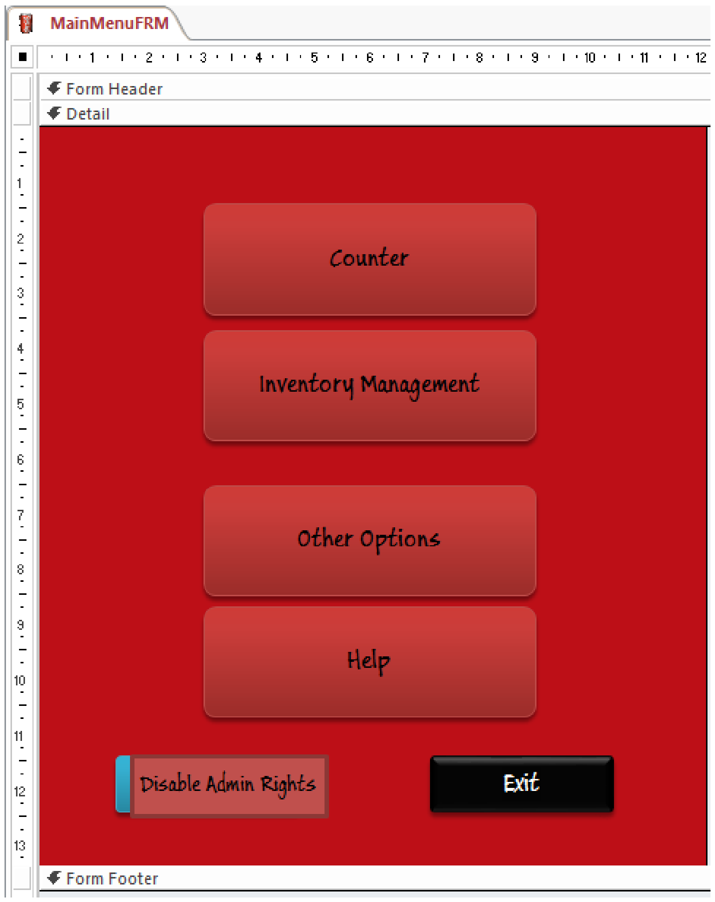
This is where all the reports are found, from all the transactions of the month to graphs on all transactions of the day.

Most of its code is switchboard but the special readme button opens a word file that should be located in the same folder as he help folder

Code

```
Dim currpath As String
currpath = CurrentProject.Path & "\Readme.docx"
Shell ("Explorer.exe " & currpath)
```

MAIN MENU FORM



FEATURES

The main menu is a special form because it has an admin status feature. It has two buttons almost in the same place, with only one visible at a given time. It gets a value from the login tables to see if the user has logged in normally or has logged in as an admin.

CODE VIEW

```
Dim mytbl As Object

Set mytbl = CurrentDb.OpenRecordset("Static Values")

With mytbl

   Modify = 1

   .Edit

   .Fields("AdminRight") = False

   .Update
```

```
End With
AdminLoginBTN.Visible = True
DoCmd.Close
DoCmd.OpenForm ("MainMenuFRM")
End Sub
```

```
Private Sub CounterBTN_Click()
DoCmd.Close
DoCmd.OpenForm ("CounterFRM")
End Sub
```

```
Private Sub Form_Load()
Dim mydb As Object
Dim mytbl As Object

Set mydb = CurrentDb
Set mytbl = mydb.OpenRecordset("Static Values")
With mytbl
  If .Fields("AdminRight") = True Then
      AdminLoginBTN.Visible = False
      AdminRightsOFFBTN.Visible = True
  Else:
  AdminLoginBTN.Visible = True
  AdminRightsOFFBTN.Visible = False
  End If
End With
End Sub
```

```
Private Sub HelpBTN_Click()

DoCmd.Close

DoCmd.OpenForm ("HelpFRM")

End Sub
```

```
Private Sub InvManBTN_Click()

DoCmd.Close

DoCmd.OpenForm ("InvMenuFRM")

End Sub
```

```
Private Sub OtherBTN_Click()

DoCmd.Close

DoCmd.OpenForm ("OtherMenuFRM")

End Sub
```

CODE EXPLANATION

The code has two sections

Button clicks, identified by the "_Click()" which function with the normal commands as the other switchboard forms.

On load: The form check if the user has logged in normally or as admin. If the user is normal the admin login button shows and the disable admin button is hidden (Not visible). The admin login button will then take the user to the admin login screen.

If the user logged in as admin the disable admin rights button will show and the admin login button will be hidden. The disable button removes admin rights and reloads the main menu form after updating the login tables.

AUTHENTICATION FORMS

These are the forms that have to deal with granting access to the system and changing variables that have to do with accessing the system. There are two basic ones although some of the forms have an admin authentication process before they open. There are two forms, the login form (normal and strict admin) and the change password form.

LOGIN FORM

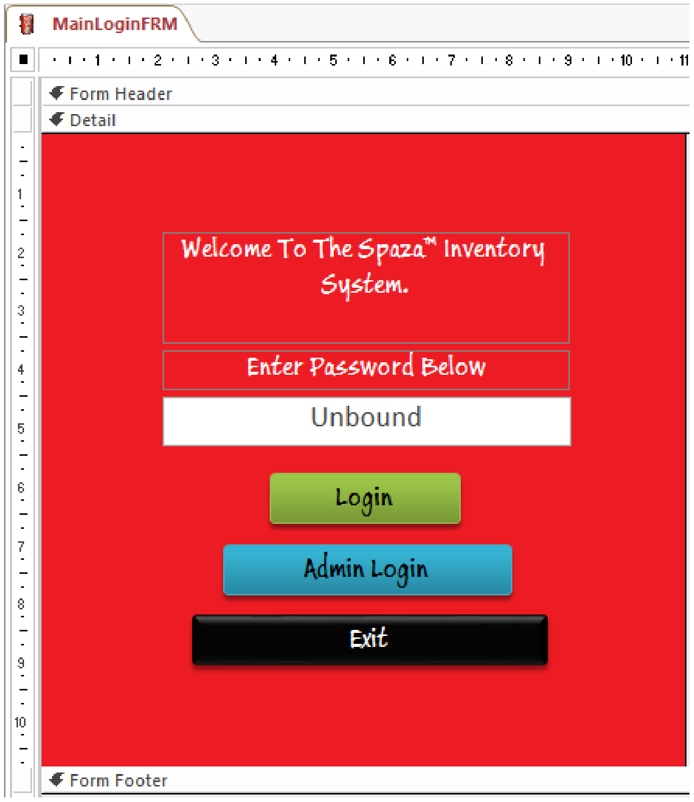
The login form is runner up in terms of code complications as I tried to make it as user friendly as possible.

It works in such a way as not to force the user to keep logging in and gives him the option to quit anytime but keeps the count of how many times the user has failed to log in.

There were a few problems faced and unusual solutions were developed.

Force Login Loop

There were difficulties with the login screen in the sense that it forced the user to login until the database locked or the user got the password correct.



Solution: Two login forms with the same design and the same code that looped each other with values stored in the static values table to produce the same outcome.

Enter Button Login

The user wanted to use the enter button to log in.

Solution: Use of the key down function was used to trigger the log in process after enter is pressed.

CODE VIEW

Private Sub AdminLoginBTN_Click()

```
Dim mydb As Object
Dim mytbl As Object
Dim approval As Boolean
Dim erratic As String
Dim pass As String

If IsNull(Me.LoginPassword) Then
   MsgBox "Please Enter A Password To Continue", vbInformation, "Password Required"
   Me.LoginPassword.SetFocus
Else: DoEvents
pass = Me.LoginPassword
Set mydb = CurrentDb
Set mytbl = mydb.OpenRecordset("Static Values")
   With mytbl
      If pass = .Fields("MasterPassword") Then
         Modify = 1
         .Edit
         .Fields("AdminRight") = True
         .Update
           Modify = 1
         .Edit
         .Fields("DatabaseLock") = False
         .Update
           Modify = 1
         .Edit
         .Fields("LoginAttempts") = 0
         .Update
         DoCmd.Close
            If DCount("*", "Reorder Query") > 0 Then
               DoCmd.OpenForm ("ReOrderLoginListFRM")
            End If
         DoCmd.OpenForm ("MainMenuFRM")
      Else
         MsgBox "Incorrect admin password - Please try again", vbInformation, "Admin Password Incorrect"
         Me.LoginPassword.SetFocus
      End If
   End With
End If
End Sub
```

Private Sub Form_Load()

```
Dim mytbl As Object
Dim mydb As Object
Dim count As Integer
Dim dbLock As Boolean

'Sets the table in use to the static values table where all the static values are stored'

Set mydb = CurrentDb
Set mytbl = mydb.OpenRecordset("Static Values")

'Removes admin rights from previous session'

With mytbl
    Modify = 1
   .Edit
   .Fields("AdminRight") = False
   .Update
End With

'This Module checks to see if the database is locked or not'

Me.LoginPassword.SetFocus
   With mytbl
      If .Fields("DatabaseLock") = True Then

'If it is locked then it will force the admin to login and unlock the database'

         DoCmd.Close
         DoCmd.OpenForm ("AdminLoginFRM")

      Else:
'If its unlocked it will open normally’
      DoEvents
      End If
   End With
End Sub
```

<u>Private Sub LoginBTN_Click()</u>

```
Dim count As Integer
Dim mydb As Object
Dim mytbl As Object
Dim approval As Boolean
Dim erratic As String
Dim dbLock As Boolean

approval = False
Me.LoginPassword.SetFocus
Set mydb = CurrentDb
Set mytbl = mydb.OpenRecordset("Static Values")
With mytbl
      dbLock = .Fields("DatabaseLock")
      count = .Fields("LoginAttempts")
         If count = 10 Or count > 10 Then
                Modify = 1
```

```
          .Edit
          .Fields("DatabaseLock") = True
          .Update
          DoCmd.Close
          DoCmd.OpenForm ("AdminLoginFRM")
      Else
        If IsNull(Me.LoginPassword) Then
          MsgBox "Please Enter A Password To Continue", vbInformation, "Password Required"
          Me.LoginPassword.SetFocus
        Else
          If Me.LoginPassword = .Fields("MainPassword") Then
            approval = True
            Modify = 1
            .Edit
            .Fields("LoginAttempts") = 0
            .Update
             Modify = 1
            .Edit
            .Fields("DatabaseLock") = False
            .Update
          Else
            count = count + 1
              If count > 10 Then
                count = 10
                .Fields("DatabaseLock") = True
              End If
            Modify = 1
            .Edit
            .Fields("LoginAttempts") = count
            .Update
            erratic = MsgBox("Error - Wrong Password Try again " & 10 - count & " Attempt(s) Left",
vbInformation, "Login Error")
          End If
      End If
  If approval = True Then
    DoCmd.Close
    If DCount("*", "Reorder Query") > 0 Then
      DoCmd.OpenForm ("ReOrderLoginListFRM")
    End If
    DoCmd.OpenForm ("MainMenuFRM")
  Else
    If count = 10 Then
      MsgBox ("Login Attempts Exceeded Enter Admin Password on next Login To Unlock")
      With mytbl
        Modify = 1
        .Edit
        .Fields("DatabaseLock") = True
        .Update
      End With
        DoCmd.Close
        DoCmd.OpenForm ("AdminLoginFRM")
    Else:
         DoCmd.Close
         DoCmd.OpenForm ("MainLoginFRM_")
      End If
   End If
   End If
End With
End Sub
```

Private Sub LoginPassword_KeyDown(KeyCode As Integer, Shift As Integer)

```
 If KeyCode = 13 Then
   Set mydb = CurrentDb
   Set mytbl = mydb.OpenRecordset("Static Values")
      With mytbl
         If .Fields("DatabaseLock") = True Then
            Me.AdminLoginBTN.SetFocus
            AdminLoginBTN_Click
         Else
            Me.LoginBTN.SetFocus
            Me.LoginPassword.SetFocus
            LoginBTN_Click
         End If
      End With
End If
End Sub
```

CODE EXPLANATION

Private Sub AdminLoginBTN_Click()

The admin button routine checks if the password matches the admin password stored in the statics value table. If the password is correct the user is allowed to the main menu and the system checks if there are any reorder products. If there are reorder product it opens the pop up form along with the main menu.

This however if the user gets it wrong doesn’t have a count so it simply continues with the error message and the reloads the form

Private Sub Form_Load()

This is on form load, the form first checks if the database is locked (attempts exceeded) if the database is locked then it shows an error message to inform the user about the situation and how to get out of it the it redirects to the admin login form.

Private Sub LoginBTN_Click()

This is the main login button click procedure.

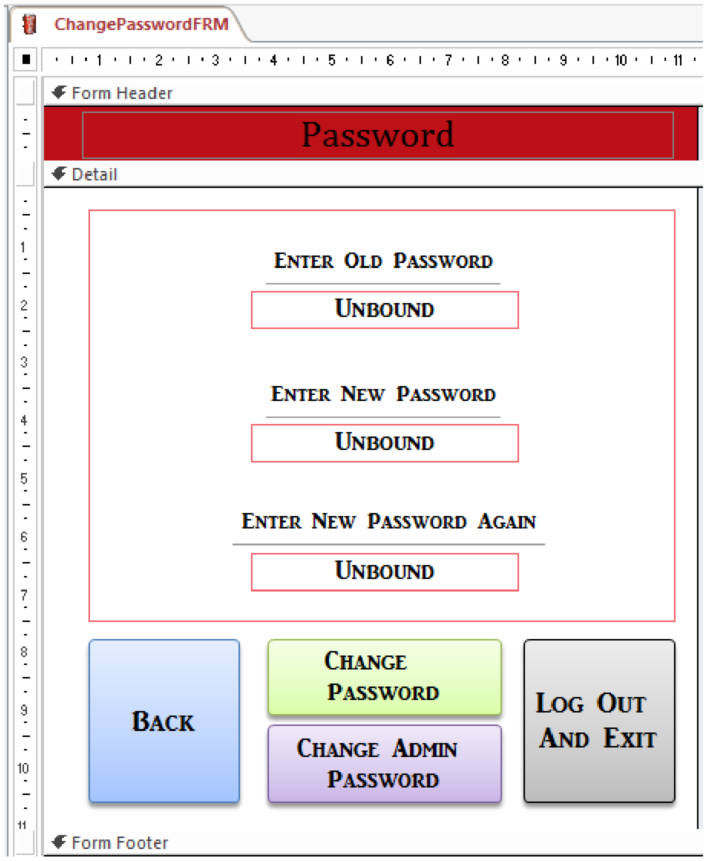
This button will start the normal login process. It checks to see if the password entered matches the password in the static values table. If the password is correct then it allows the user to the main menu and checks if there are any reorder products. If there are reorder product it opens the pop up form along with the main menu.

Private Sub LoginPassword_KeyDown(KeyCode As Integer, Shift As Integer)

This subroutine is called upon when the enter button is pressed. It checks if the database is locked and then initiates the login button process.

CHANGE PASSWORD FORM

This form is used to change passwords. To change passwords of course the admin rights are needed and extreme validation checks to see if the user really wants to change their password.



This form had unbound textboxes and has a straight forward process so not many problems were encountered when it was beng developed.

CODE VIEW

Private Sub BackButtonBTN_Click()

```
DoCmd.Close
DoCmd.OpenForm ("OtherMenuFRM")
End Sub
```

Private Sub ChangeAdminBTN_Click()

```
Dim mytbl As Object
Dim dupcheck As Boolean
Dim checkcheck As String
Set mytbl = CurrentDb.OpenRecordset("Static Values")
If Me.NPAgainTXT = Me.NewPasswordTXT Then dupcheck = True
With mytbl
If Me.OldPassTXT = .Fields("MasterPassword") Then
```

```
  If dupcheck = True Then
    checkcheck = inputbox("To complete the process please enter the new admin password again", "Admin
Password Clarification")
    If checkcheck = Me.NPAgainTXT Then
      Modify = 1
      .Edit
      .Fields("MasterPassword") = Me.NewPasswordTXT
      .Update
      MsgBox "Your admin password has been changed!", vbInformation, "Password Change Successful"
       Me.OldPassTXT = ""
         Me.NewPasswordTXT = ""
    Me.NPAgainTXT = ""
    Else: MsgBox "Your new admin password doesnt match the one you typed into the form please press the
change admin password button and try again again!" _
      , vbInformation, "New Password Inconsistency"
    End If
  Else
    MsgBox "Your new password is inconsistent! Type in your new passwords again", vbInformation, "New
Password Error"
    Me.NPAgainTXT.SetFocus
  End If
Else: MsgBox "Your password is incorrect! Please try again", vbInformation, "Password Error"
Me.NewPasswordTXT = ""
Me.OldPassTXT = ""
Me.NPAgainTXT = ""
Me.OldPassTXT.SetFocus
End If
End With
End Sub
```

<u>Private Sub ChangePassBTN_Click()</u>

```
Dim mytbl As Object
Dim dupcheck As Boolean
Set mytbl = CurrentDb.OpenRecordset("Static Values")
If Me.NPAgainTXT = Me.NewPasswordTXT Then dupcheck = True
With mytbl
If Me.OldPassTXT = .Fields("MainPassword") Then
  If dupcheck = True Then
    Modify = 1
    .Edit
    .Fields("MainPassword") = Me.NewPasswordTXT
    .Update
    MsgBox "Your main password has been changed!", vbInformation, "Password Change Successful"
    Me.OldPassTXT = ""
    Me.NewPasswordTXT = ""
    Me.NPAgainTXT = ""
  Else
    MsgBox "Your new password is inconsistent! Type in your new passwords again", vbInformation, "New
Password Error"
    Me.NPAgainTXT = ""
    Me.NPAgainTXT.SetFocus
```

```
   End If
Else: MsgBox "Your password is incorrect! Please try again", vbInformation, "Password Error"
Me.NewPasswordTXT = ""
Me.OldPassTXT = ""
Me.NPAgainTXT = ""
Me.OldPassTXT.SetFocus
End If
End With
End Sub
```

Private Sub Form_Load()

```
Dim stattbl As Object
Set stattbl = CurrentDb.OpenRecordset("Static Values")
With stattbl
If .Fields("AdminRight").Value = False Then
   DoCmd.Close
   MsgBox "You dont have permission to change passwords!!", vbInformation, "Admin Rights Required"
   DoCmd.OpenForm ("MainMenuFRM")
End If
End With
End Sub
```

CODE EXPLANATION

Private Sub BackButtonBTN_Click()

This is a normal switch button (Glossary)

Private Sub ChangeAdminBTN_Click()

This is initiated when the change admin password button is clicked it first checks if the old admin password matches the current master password and only after that has been approved it then checks if the user has typed in both of the new passwords consistently and then will send an input box for the user to type in the new admin password again as it is very important and should not be forgotten. Only after these validation processes will it start the changing admin password process with the static values table.

If however something is wrong the appropriate message boxes will be displayed

Private Sub ChangePassBTN_Click()

This is initiated when the change password button is clicked it first checks if the old password matches the current password and only after that has been approved it then checks if the user has typed in both of the new passwords consistently and only then after will it start the changing process.

If however something is wrong the appropriate message boxes will be displayed

Private Sub Form_Load()

On form load the system checks if the user has admin rights. If the user does then events continue as normal. If not the form redirects the user to the main menu after informing the user about what is going on.

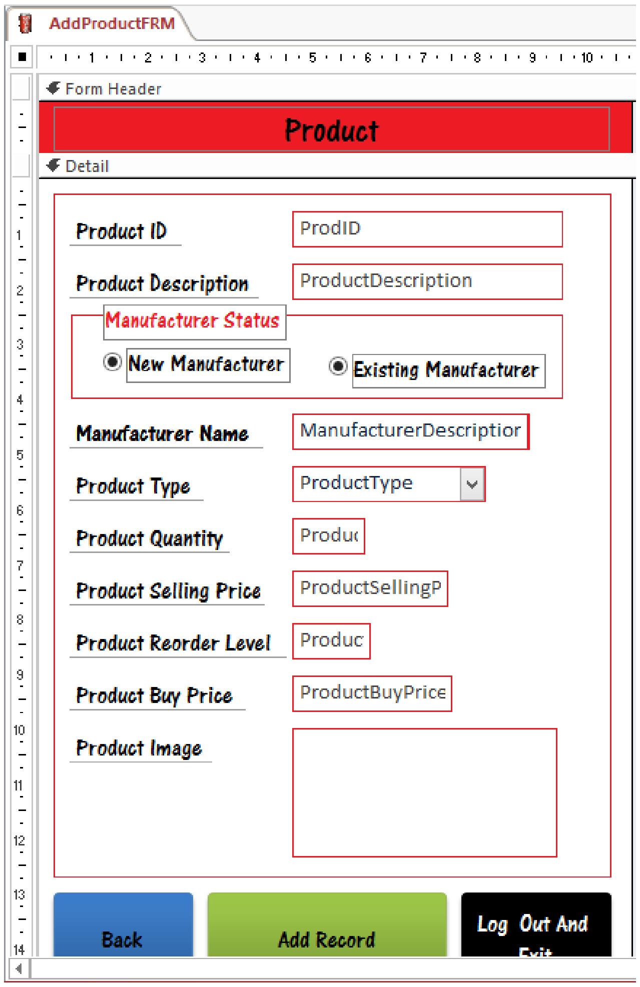
DATA MANAGEMENT FORMS

These are the forms that add, modify and delete records from the tables. The rest of the forms are classified as data management as they fit under this criteria.

ADD PRODUCT FORM

There was a problem with the table to do with the manufacturers. I wanted to give the user an option of adding a new manufacturer without making an add manufacturer form so I made it an option.

Radio buttons (Option Buttons) were used to achieve the effect of allowing the user to add a new manufacturer to the manufacturer table and then set the manufacturer of the current product to the one that was just added by linking the manufacturer description field straight from the manufacturer table. The radio buttons are used as a switch in actual fact all they do is show the appropriate objects and hide the inappropriate objects depending on the user option.



CODE VIEW

Private Sub AddRecBTN_Click()

```
DoCmd.RunCommand acCmdSaveRecord
MsgBox "Your record has been saved", vbInformation, "Request Succesful"
DoCmd.GoToRecord , "", acNewRec
End Sub
```

Private Sub BackBTN_Click()

```
DoCmd.Close
DoCmd.OpenForm ("ModDBFRM")
End Sub
```

Private Sub ExitBTN_Click()

```
DoCmd.Close
End Sub
```

Private Sub ExManuOpt_Click()

```
NewManuOpt = False
ExManuOpt = True
ManufacturerDescription.Visible = False
ManufacturerID.Visible = True
End Sub
```

Private Sub Form_Load()

```
ExManuOpt_Click
DoCmd.GoToRecord , "", acNewRec
End Sub
```

Private Sub NewManuOpt_Click()

```
NewManuOpt = True
ExManuOpt = False
ManufacturerDescription.Visible = True
ManufacturerID.Visible = False
End Sub
```

CODE EXPLANATION

Private Sub AddRecBTN_Click()

This button takes everything in the boxes and then adds the details to the product table. If a new manufacturer was added its details will be saved to the manufacturer table

Private Sub BackBTN_Click()

This is a normal switch button (Glossary)

Private Sub ExitBTN_Click()
This is a normal switch button (Glossary)

Private Sub ExManuOpt_Click()
This hides the textbox for manufacturer description and adds a combo box with the list of existing manufacturers

Private Sub Form_Load()
On form load the system checks if sets the record to new. This means that instead of showing the first or last records in the products table it just shows the fields empty awaiting data

Private Sub NewManuOpt_Click()
This hides the combo box for existing manufacturers and adds a text box to allow the user to type in the description of their new manufacturer.

MODIFY PRODUCT FORM

This is where the user can view his products record by record, edit the details and delete the whole product record.

This is quite a simple form in terms of programming is concerned because it was directly linked to the product table allowing the programmer to make use of the "docmd" commands.

CODE VIEW

Private Sub BackBTN_Click()
```
DoCmd.Close
DoCmd.OpenForm ("ModDBFRM")
End Sub
```

Private Sub DeleteRecBTN_Click()
```
DoCmd.RunCommand acCmdDeleteRecord
End Sub
```

Private Sub ExitBTN_Click()
```
DoCmd.Close
End Sub
```

Private Sub FindRecordBTN_Click()
```
Dim mytbl As Object
Dim searchid As Integer
Dim positioncount As Integer
Dim found As Boolean
Set mytbl = CurrentDb.OpenRecordset("Product")
searchid = inputbox("Enter product ID to find record", "Search ID")
With mytbl
found = False
.MoveFirst
positioncount = 1
  Do
    If searchid = .Fields("ProdID") Then
    found = True
      DoCmd.GoToRecord , "", acGoTo, positioncount
    End If
    positioncount = positioncount + 1
    .MoveNext
  Loop Until .EOF
End With
If found = False Then
  MsgBox "The product ID does not exist!! Please check in the view database to get the desired ID",
vbInformation, "Invalid ID"
End If
End Sub
```

<u>Private Sub FirstRecBTN_Click()</u>

```
DoCmd.GoToRecord , "", acFirst
End Sub
```

<u>Private Sub LastRecBTN_Click()</u>

```
DoCmd.GoToRecord , "", acLast
End Sub
```

<u>Private Sub NextRecBTN_Click()</u>

```
DoCmd.GoToRecord , "", acNext
End Sub
```

<u>Private Sub PrevRecBTN_Click()</u>

```
DoCmd.GoToRecord , "", acPrevious
End Sub
```

<u>Private Sub SaveBTN_Click()</u>

```
DoCmd.RunCommand acCmdSaveRecord
MsgBox "Your changes have been saved", vbInformation, "Request Successful"
End Sub
```

CODE EXPLANATION

Private Sub BackBTN_Click()
This is a normal switch button (Glossary)

Private Sub DeleteRecBTN_Click()
This code deletes the product that is being viewed when the button was pressed

Private Sub ExitBTN_Click()
This is a normal switch button (Glossary)

Private Sub FindRecordBTN_Click()
This starts the find button using the search ID typed into the input box that appears when the button is pressed.

Because it's difficult to search and retrieve details on a product. The use of a count function that counts how many positions the system has to loop to get to the record and then using the record position the product details are fetched using the "go to" function and the count as a product position to go to.

Private Sub FirstRecBTN_Click()
This goes to and displays the first record of the table

Private Sub LastRecBTN_Click()
This goes to and displays the Last record of the table

Private Sub NextRecBTN_Click()
This goes to the next record and displays from the table

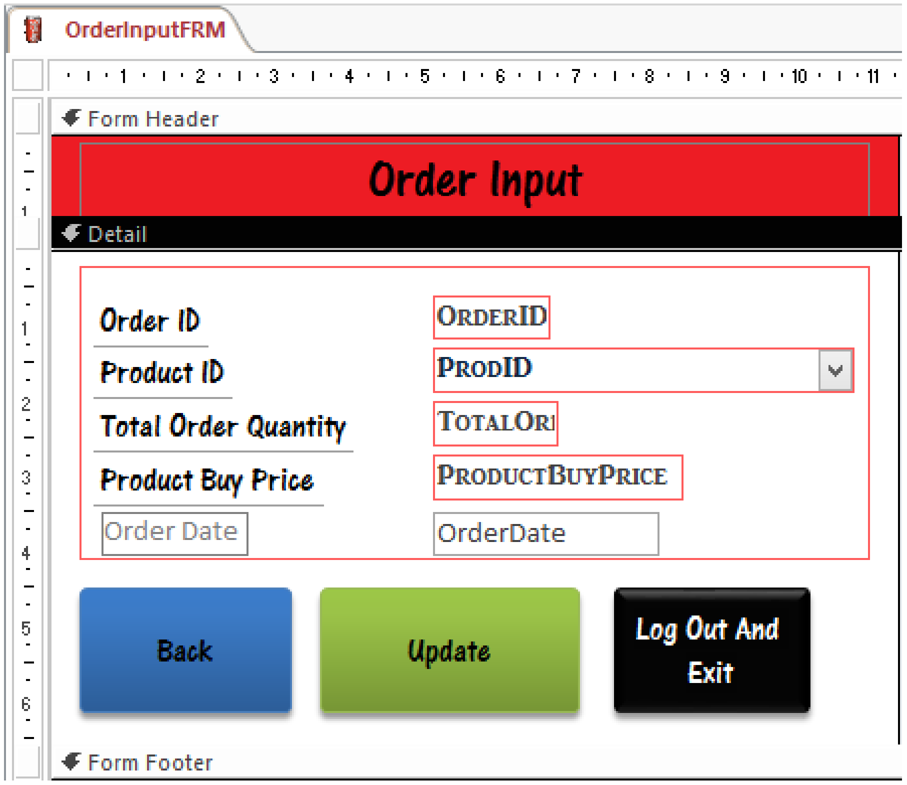
Private Sub PrevRecBTN_Click()
This goes to the previous record and displays from the table

Private Sub SaveBTN_Click()
The save button saves changes made to the prouct being viewed when clicked

ORDER INPUT FORM

This is the form used to input orders and update stock.

The order date is hidden and automatically set to reduce user work and validation purposes and to keep it as accurate as possible.



CODE VIEW

Private Sub BackBTN_Click()

```
DoCmd.Close
Dim stattbl As Object
Set stattbl = CurrentDb.OpenRecordset("Static Values")
With stattbl
If .Fields("AdminRight").Value = False Then
   DoCmd.OpenForm ("MainMenuFRM")
Else
DoCmd.OpenForm ("ModDBFRM")
End If
End With
End Sub
```

Private Sub Form_Load()

```
DoCmd.GoToRecord , "", acNewRec
End Sub
```

```
Private Sub UpdateBTN_Click()
Dim Quantity As Integer
Dim mytbl As Object
Dim MyProduct As Integer
Me.OrderDate = Date
MyProduct = Me.ProdID
Quantity = Me.TotalOrderQuantity
Set mytbl = CurrentDb.OpenRecordset("Product")
With mytbl
   .MoveFirst
      Do
         If MyProduct = .Fields("ProdID") Then
         Modify = 1
         .Edit
         .Fields("ProductQuantity") = .Fields("ProductQuantity") + Quantity
         .Update
         DoCmd.GoToRecord , "", acNewRec
         Exit Do
         End If
         .MoveNext
      Loop Until .EOF
End With
MsgBox "Order input complete, Database updated!", vbInformation, "Request Succesful"
End Sub
```

CODE EXPLANATION

Private Sub BackBTN_Click()

This is a normal switch button (Glossary)

Although it checks if the user has admin right if the user has admin rights he returns to the data modification switchboard and if not the user is returned to the main menu.

Private Sub Form_Load()

On form load the form creates a blank record in the order table.

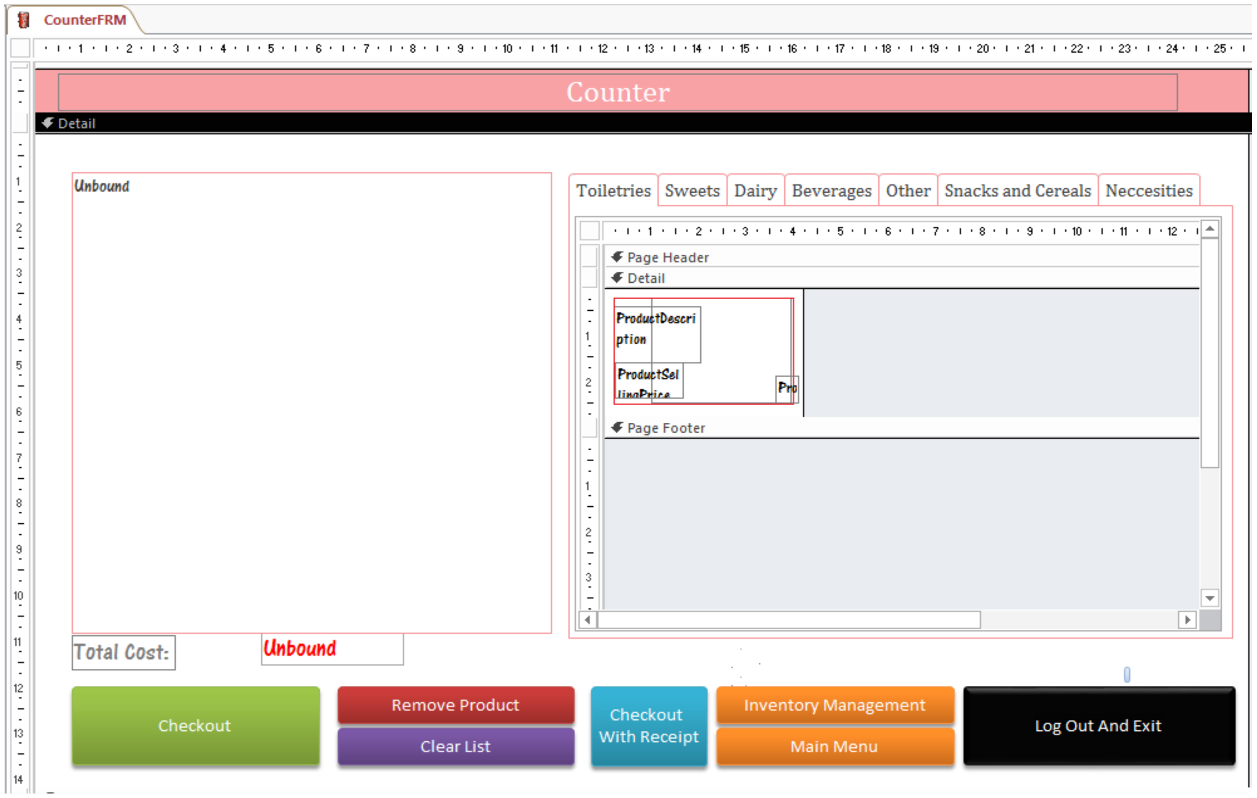
Private Sub UpdateBTN_Click()

The update button saves record and resets the form. While the record is being saved the form then edits the quantity in the product table to add the ordered tables

This is the form that has the most complicated programing structure.

FEATURES

- Labels with pictures which are used as icons to create that GUI interface.
- A list with three columns and permanent headings.
- A tab control is used to classify product labels by type.
- Hidden text Boxes and labels directly linked to the transaction and sales table.
- A function to produce a receipt when required.
- A subtotal box that is recalculated when something is added or removed from the list.
- A hidden button that initiates the "add to list" function.



CODE VIEW

```
Option Explicit
Public ProductPos As Integer
Public BoughtQuantity As Integer
Public costsum As Currency
```

```
'This sub routine searches the product table for the product(s) involved in the transaction and updates the
stock
Public Sub QuantitySave(ByVal ident As Integer, ByVal quanti As Integer)
Dim prodtbl As Object
Dim mydb As Object
Dim final As Integer
Set mydb = CurrentDb
Set prodtbl = mydb.OpenRecordset("Product")
With prodtbl
  .MoveFirst
    Do
    'the sub routine uses the ID to search the product table
      If ident = .Fields("ProdID") Then
        final = .Fields("ProductQuantity") - quanti
        prodtbl.Edit
        prodtbl("ProductQuantity").Value = final
        prodtbl.Update
      End If
    .MoveNext
    Loop Until .EOF
End With
End Sub
```

```
' This sub routine is the one that adds a data item selected from the icon menu to the list
Public Sub AddToList_Click()
Dim mytbl As Object
Dim prodtbl As Object
Dim Description As String
Dim count As Integer
Dim countst As String
Dim CostCurrency As String
Dim costa As Currency
Dim currentindex As Integer
Dim onlist As Boolean
Set mytbl = CurrentDb.OpenRecordset("Static Values")
Set prodtbl = CurrentDb.OpenRecordset("Product")
onlist = False

With mytbl
'Setting the description to the temporary container
  Description = .Fields("CurrentProdDesc")
End With
'Check to see if it exists on the list
For currentindex = 1 To Me.ProductReceiptLST.ListCount - 1
  If Description = Me.ProductReceiptLST.Column(0, currentindex) Then
      With prodtbl
        .MoveFirst
  'Searching for the product Records using he description and then fetches the details on the product(s)
involved and updates the quntity only
        Do
          If Description = .Fields("ProductDescription") Then
```

```
                    CostCurrency = .Fields("ProductSellingPrice")
                    costa = .Fields("ProductSellingPrice")
                    count = Me.ProductReceiptLST.Column(2, currentindex)
                    count = count + 1
                  End If
                  .MoveNext
                Loop Until .EOF
'formating the data because the list box only accepts string data
                countst = Format(count, "General Number")
                CostCurrency = Format(CostCurrency, "Currency")
                Me.ProductReceiptLST.AddItem Description & "," & CostCurrency & "," & countst, currentindex
                Me.ProductReceiptLST.RemoveItem (currentindex + 1)
' updating of sub totals
                costsum = costsum + costa
                TotalMoneyTXT = costsum
                onlist = True
            End With
    End If
Next
'if it isnt on the list then it adds the item to the list
If onlist = False Then
With prodtbl
    .MoveFirst
    'Searching for the product Records
    Do
        If Description = .Fields("ProductDescription") Then
            CostCurrency = .Fields("ProductSellingPrice")
            costa = .Fields("ProductSellingPrice")
            count = 1
        End If
        .MoveNext
    Loop Until .EOF
'formating to string data and adding to list
    countst = Format(count, "General Number")
    CostCurrency = Format(CostCurrency, "Currency")
    Me.ProductReceiptLST.AddItem Description & "," & CostCurrency & "," & countst
'Updating sub totals
    costsum = costsum + costa
    TotalMoneyTXT = costsum
End With
End If
End Sub
```

<u>Private Sub CheckoutBTN_Click()</u>

```
Dim prodtbl As Object
Dim saletbl As Object
Dim transtbl As Object
Dim i As Integer
Dim currentid As Integer
Dim currentprod As String
Dim currentprodid As Integer
Dim remposi As Integer
Dim amountpaid As Currency
DoCmd.GoToRecord , "", acNewRec
'check to see if the list box is empty
If Me.SubTotal = 0 Or Me.TotalMoneyTXT = 0 Then
   MsgBox "Please input items to check out", vbInformation, "Empty list"
Else
Set prodtbl = CurrentDb.OpenRecordset("Product")
'if  the list box isnt empty then it starts extracting data from the list
'sets the imaginary fields using data from the list
   Me.TransactionDate = Date
   Me.TransactionTime = Time
   Me.SubTotal = Me.TotalMoneyTXT
   currentid = Me.TransactionID
   DoCmd.RunCommand acCmdSaveRecord
'The input box helps to calculate change
   amountpaid = inputbox("Please Enter Amount paid")
'Updating the sales table using the same transaction id for each item on the list
   For i = 1 To Me.ProductReceiptLST.ListCount - 1
      DoCmd.GoToRecord , "", acNewRec
      Me.TransactionID_Sales = currentid
      currentprod = Me.ProductReceiptLST.Column(0, i)
      Me.QuantityBought = Me.ProductReceiptLST.Column(2, i)
      BoughtQuantity = Me.ProductReceiptLST.Column(2, i)
         'Using the data from the sales list to look for extra prouct details
         With prodtbl
            .MoveFirst
            Do
               If currentprod = .Fields("ProductDescription") Then
                  currentprodid = .Fields("ProdID")
                  Me.ProdID = currentprodid
                  'Calls upon the QuantitySave subroutine that then updates the stock levels of each product
                  Form_CounterFRM.QuantitySave currentprodid, BoughtQuantity
               End If
               .MoveNext
            Loop Until .EOF
         End With
      DoCmd.RunCommand acCmdSaveRecord
   Next
'Shows the admin the change due from calculations with the totals and input that was asked for earlier
'Shows next customer to signal that the transaction was complete
MsgBox Format(amountpaid - TotalMoneyTXT, "currency"), vbInformation, "The Change Due"
MsgBox "Next Customer", vbInformation, "Thank You"
'Clearing The List Box with the subroutine
Call ClearList
End If
'Going the a new trasaction and sale record ready for the next customer
DoCmd.GoToRecord , "", acNewRec
End Sub
```

<u>Private Sub ClearBTN_Click()</u>

```
'Calls the clear list subroutine
Call ClearList
End Sub
```

Public Sub ClearList()

```
' the clear list subroutine
Dim clearindex As Integer
Dim remposi As Integer
'uses the max index to clear the list by popping of the last object each cycle
'Until the headings are left
Do Until clearindex = Me.ProductReceiptLST.ListCount - 1
remposi = (Me.ProductReceiptLST.ListCount - 1)
Me.ProductReceiptLST.RemoveItem (remposi)
Loop
'Reseting sub totals
Me.TotalMoneyTXT = 0
costsum = 0
End Sub
```

Private Sub Form_Load()

```
'Clearing previous transactions on load and clearing the list
DoCmd.GoToRecord , "", acNewRec
Call ClearList
End Sub
```

Private Sub InvManBTN_Click()

```
DoCmd.Close
DoCmd.OpenForm ("InvMenuFRM")
End Sub
```

Private Sub MainMenuBTN_Click()

```
DoCmd.Close
DoCmd.OpenForm ("MainMenuFRM")
End Sub
```

Private Sub ReceiptCheckoutBTN_Click()

```
CheckoutBTN_Click
DoCmd.OpenReport ("LastTransactReceiptRP")
End Sub
```

Private Sub RemoveItemBTN_Click()

```
'The subroutine that removes only the item selected
Dim TempSelected As Integer
Dim moneyreduced As Currency
Dim quantum As Integer
TempSelected = Me.ProductReceiptLST.ListIndex + 1
```

```
If TempSelected = 0 Then
   MsgBox "No item is selected to be removed", vbInformation, "Selection error"
Else
'It removes the item from the list and uses its details to recalculate the sub totals
   moneyreduced = Me.ProductReceiptLST.Column(1, TempSelected)
   quantum = Me.ProductReceiptLST.Column(2, TempSelected)
   moneyreduced = moneyreduced * quantum
   Me.TotalMoneyTXT = Me.TotalMoneyTXT - moneyreduced
   costsum = costsum - moneyreduced
   Me.ProductReceiptLST.RemoveItem (TempSelected)
End If
End Sub
```

CODE EXPLANATION

Public Sub QuantitySave(ByVal ident As Integer, ByVal quanti As Integer)

This function looks for the product(s) involved in the transaction and then updates the stock level depending on how much is bought and the final value is stored in the product table.

Public Sub AddToList_Click()

This is the small hidden button that is used with the value of the current product description in the static values table. This adds the appropriate data to the appropriate columns in the list and first checks to see if the item is nonexistent on the list using a search function. If the product exists then the quantity is just updated as well as the subtotal but without a new entry of the same product.

Private Sub CheckoutBTN_Click()

This button triggers the checkout process

The checkout process:

- Makes a new transaction file
- Sets the subtotal, Time and date from the system clock and the total calculated textbox.
- Uses the transaction number for the sales of the individual products.
- The list is designed to work as a stack and the checkout process uses a for next loop to go through every single entry
- It Calls quantity save while passing on the quantity values and the product values to update the stock
- Uses the same Transaction number and updates the sales table for each product till the list finishes and then clears the list.

Private Sub ClearBTN_Click()

The clear list button calls upon the clear list function.

Public Sub ClearList()

The call list subroutine:

- Uses a stack function to pop off items one by one using a “for next” loop until the list is empty.

Private Sub Form_Load()
On form load a new transaction and sales record is initiated and the form is cleared to make sure.

Private Sub InvManBTN_Click()
This is a normal switchboard button(Glossary)

Private Sub MainMenuBTN_Click()
This is a normal switchboard button(Glossary)

Private Sub ReceiptCheckoutBTN_Click()
This processes transaction the same way as normal checkout but produces the most recent transaction report to the default printer when it’s done processing.

Private Sub RemoveItemBTN_Click()
By using form index function to get the selected list item index the user can remove the selected data from the check-out list in one click.

VARIABLE LIST

Because the list had a lot of variables being passed around it felt appropriate to explain what role each of them play in the code.

PUBLIC VARIABLES

Public BoughtQuantity As Integer: This allows the amount of products bought for each item to be passed and changed with each sub routine.
Public costsum As Currency: This variable edits/contains the subtotal as the processes continue

Public Sub QuantitySave(ByVal ident As Integer, ByVal quanti As Integer)

Dim prodtbl As Object: Contains the current table being manipulated/used.
Dim mydb As Object: Contains the current database being manipulated/used.
Dim final As Integer: Contains the final quantity for stock after the processing.

Public Sub AddToList_Click()

Dim prodtbl As Object: Contains the current table being manipulated/used.
Dim mydb As Object: Contains the current database being manipulated/used.
Dim Description As String: Contains the current description of the product in the list being evaluated.
Dim count As Integer: Keeps count of the existing product quantities on the lists.
Dim countst As String: keeps the count in string for the list
Dim CostCurrency As String: Contains the string format of the cost to be displayed on the list.
Dim costa As Currency: Contains the selling price to edit the sub total
Dim currentindex As Integer: Contains the current list index being evaluated to see if the element exists on the list or is used to add products to the last place
Dim onlist As Boolean: Is yes if the item being added exists on the list already and isn't when it's not

Private Sub CheckoutBTN_Click()

Dim prodtbl As Object: Contains the product table being manipulated/used.
Dim saletbl As Object: sales table being manipulated/used.
Dim transtbl As Object: Contains the transaction table being manipulated/used.
Dim i As Integer:
Dim currentid As Integer: Stores the transaction Id and reproduces it for the number of sales
Dim currentprod As String: Stores the string value of the product description being used
Dim currentprodid As Integer: Stores the product ID number
Dim remposi As Integer: Stores the record position of the product being evaluated.
Dim amountpaid As Currency: This is the amount the customer pays simple for the purpose of helping them to calculate change

Public Sub ClearList()

Dim clearindex As Integer: Stores the list length as a given time
Dim remposi As Integer: Stores the current index in the loop being removed.

Private Sub ReceiptCheckoutBTN_Click()

Has the same list of variables as the normal check out process

Private Sub RemoveItemBTN_Click()

Dim TempSelected As Integer: Stores the integer value of the selected index.
Dim moneyreduced As Currency: Helps to calculate the sub total after removing an item.
Dim quantum As Integer: Stores the instances of a certain product before they are removed.

QUERY DESIGN EXPLANATION AND CODE

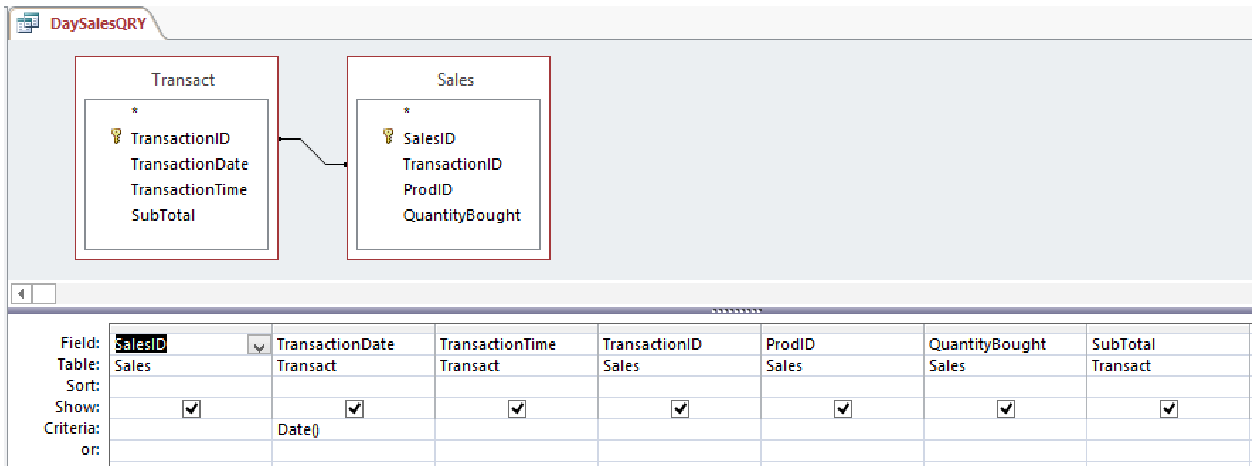
Regarding the fact that all reports were derived directly from queries it makes it appropriate to see how the queries were used to filter out the necessary data.

THE SALES QUERIES

There are 4 sales queries that keep track of the whole transaction sales records. It combines the two tables and then filters the out for a time period

DAY SALES QUERY

This filters out all the sales that occur on the particular day.



DaySalesQRY

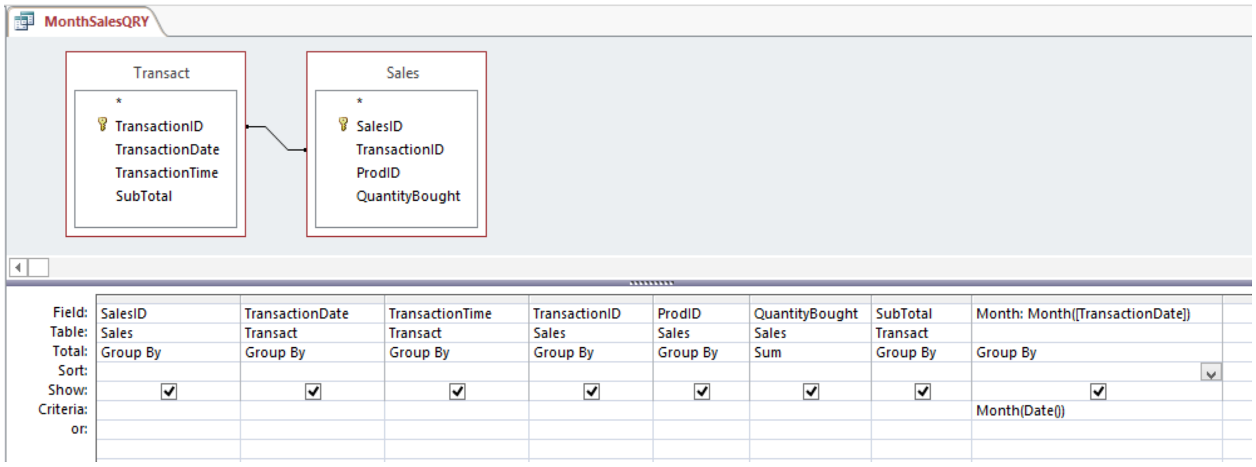
Transact: *, TransactionID, TransactionDate, TransactionTime, SubTotal

Sales: *, SalesID, TransactionID, ProdID, QuantityBought

Field:	SalesID	TransactionDate	TransactionTime	TransactionID	ProdID	QuantityBought	SubTotal
Table:	Sales	Transact	Transact	Sales	Sales	Sales	Transact
Sort:							
Show:	✓	✓	✓	✓	✓	✓	✓
Criteria:		Date()					
or:							

MONTHLY SALES QUERY

This filters out all the sales that occur during the current month.



MonthSalesQRY

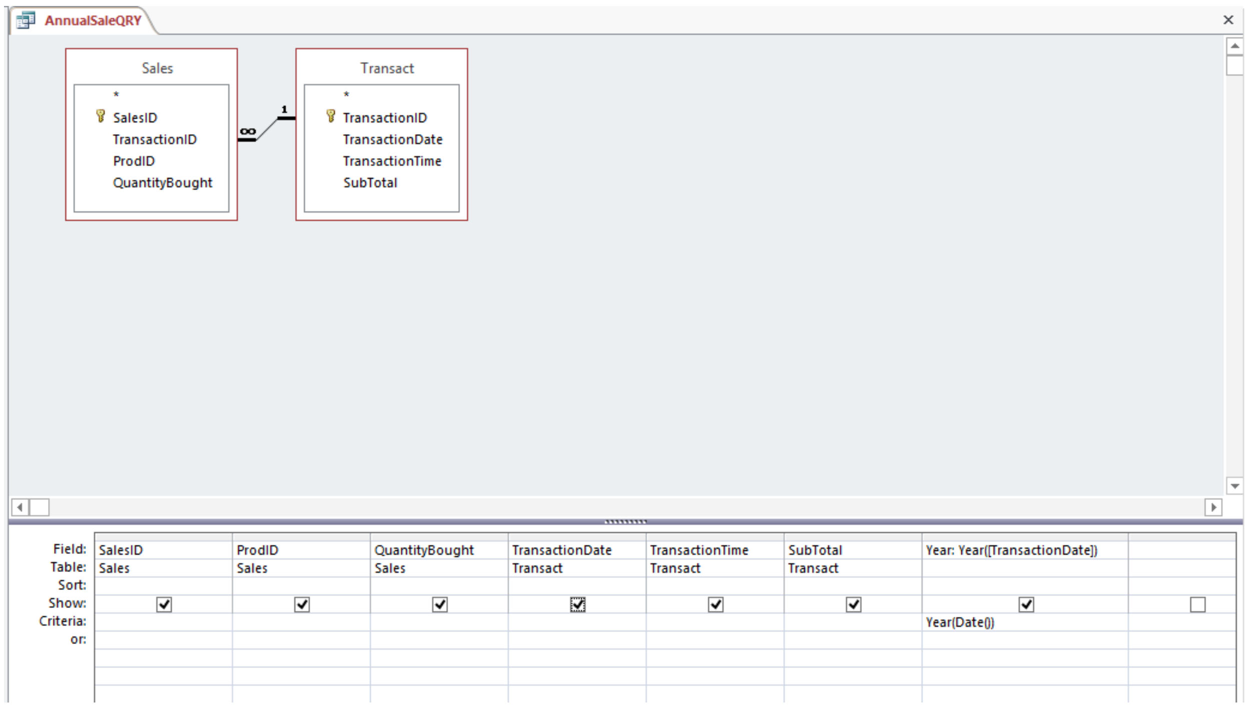
Transact: *, TransactionID, TransactionDate, TransactionTime, SubTotal

Sales: *, SalesID, TransactionID, ProdID, QuantityBought

Field:	SalesID	TransactionDate	TransactionTime	TransactionID	ProdID	QuantityBought	SubTotal	Month: Month([TransactionDate])
Table:	Sales	Transact	Transact	Sales	Sales	Sales	Transact	
Total:	Group By	Group By	Group By	Group By	Group By	Sum	Group By	Group By
Sort:								
Show:	✓	✓	✓	✓	✓	✓	✓	✓
Criteria:								Month(Date())
or:								

ANNUAL QUERY

This filters out all the sales that occur through the course of the current year.



AnnualSaleQRY

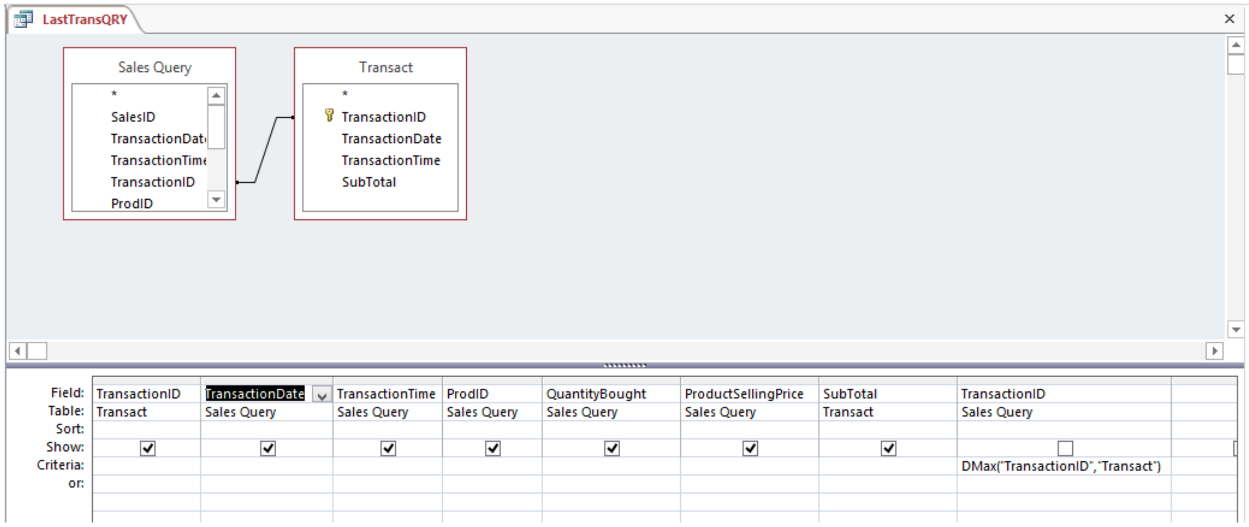
Sales: *, SalesID, TransactionID, ProdID, QuantityBought

Transact: *, TransactionID, TransactionDate, TransactionTime, SubTotal

Field:	SalesID	ProdID	QuantityBought	TransactionDate	TransactionTime	SubTotal	Year: Year([TransactionDate])	
Table:	Sales	Sales	Sales	Transact	Transact	Transact		
Sort:								
Show:	✓	✓	✓	✓	✓	✓	✓	☐
Criteria:							Year(Date())	
or:								

LAST TRANSACTION QUERY

This filters out all the sales that occurred during the last transaction which is also used to filter receipt details after a transaction.



LastTransQRY

Sales Query: *, SalesID, TransactionDat, TransactionTime, TransactionID, ProdID

Transact: *, TransactionID, TransactionDate, TransactionTime, SubTotal

Field:	TransactionID	TransactionDate	TransactionTime	ProdID	QuantityBought	ProductSellingPrice	SubTotal	TransactionID
Table:	Transact	Sales Query	Sales Query	Sales Query	Sales Query	Sales Query	Transact	Sales Query
Sort:								
Show:	✓	✓	✓	✓	✓	✓	✓	☐
Criteria:								DMax("TransactionID","Transact")
or:								

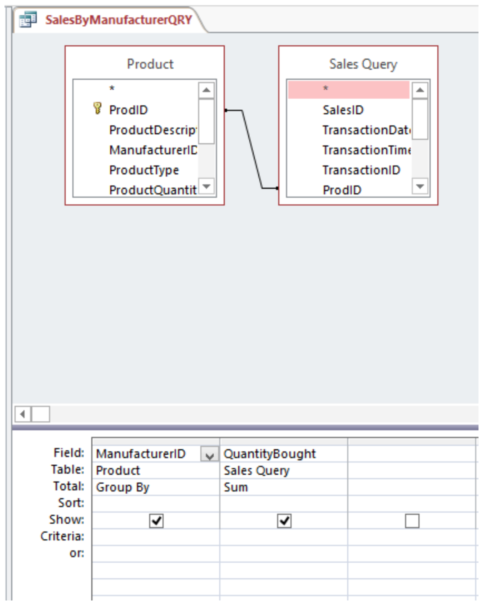
This particular query is close to my heart in the sense that it analyses the transaction id based on the fact that it's an auto-number means the largest one will automatically be the most recent. This allows the user then to make a transaction/ Sales receipt at hand.

GENERAL QUERIES

These queries don't have a special function all they do is filter out data and groups it by the criteria.

SALES BY MANUFACTURER

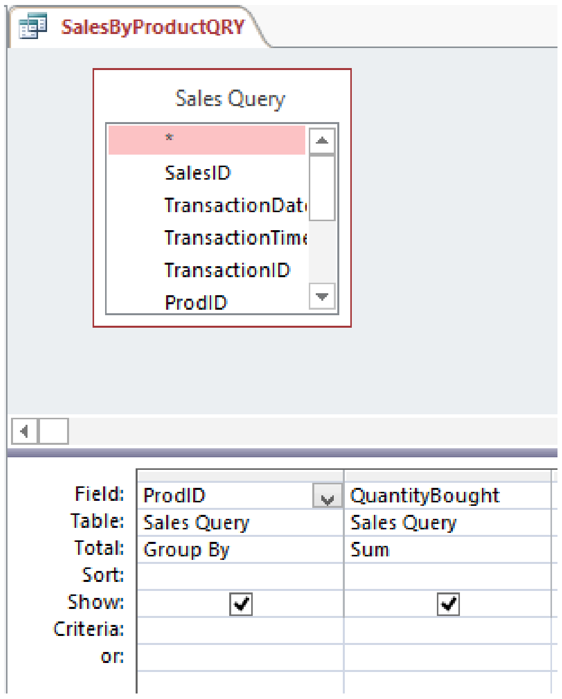
This keeps track of the sales of the individual manufacturers



Field:	ManufacturerID	QuantityBought	
Table:	Product	Sales Query	
Total:	Group By	Sum	
Sort:			
Show:	✓	✓	☐
Criteria:			
or:			

SALES BY PRODUCT NAME

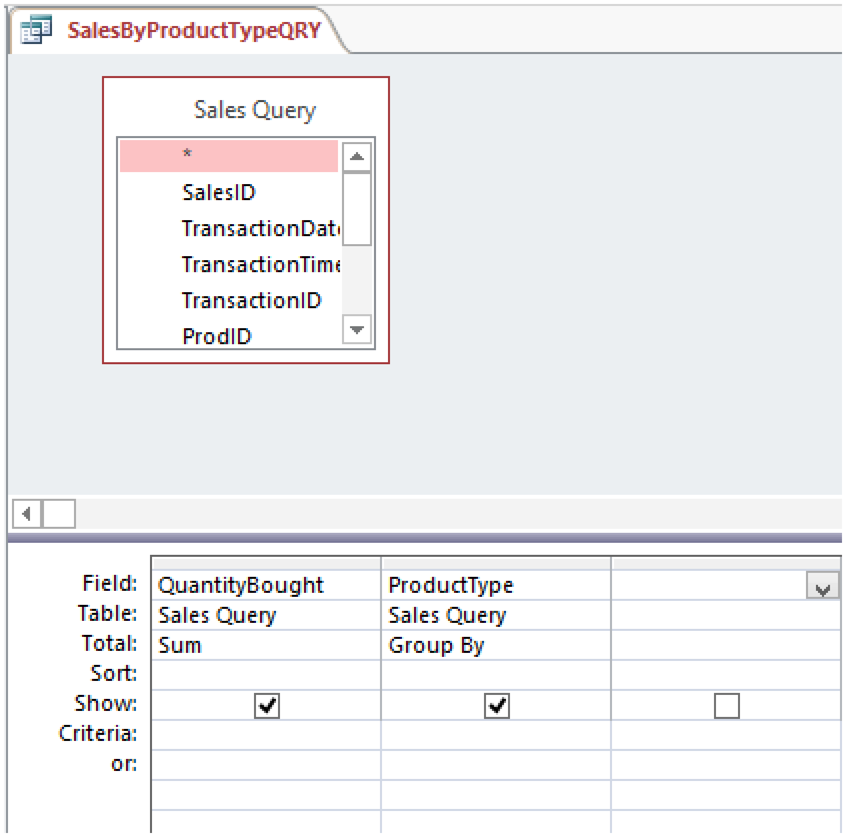
This keeps track of the sales of the individual products.



Field:	ProdID	QuantityBought
Table:	Sales Query	Sales Query
Total:	Group By	Sum
Sort:		
Show:	✓	✓
Criteria:		
or:		

SALES BY PRODUCT TYPE

This keeps track of the sales of the individual product types.



SalesByProductTypeQRY

Sales Query

*
SalesID
TransactionDat
TransactionTime
TransactionID
ProdID

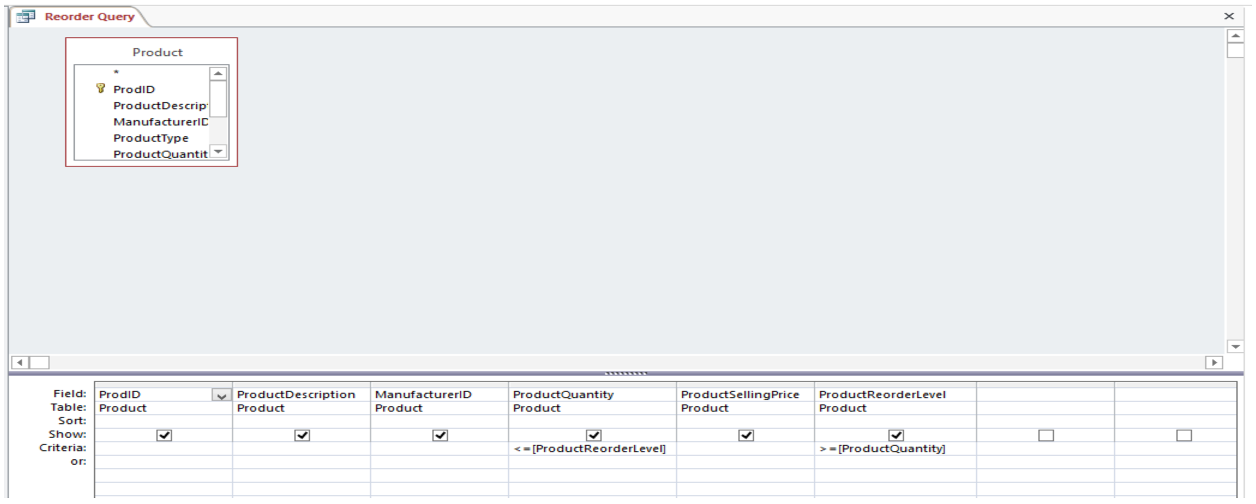
Field:	QuantityBought	ProductType	
Table:	Sales Query	Sales Query	
Total:	Sum	Group By	
Sort:			
Show:	✓	✓	☐
Criteria:			
or:			

OTHER QUERIES

The rest of the queries that didn't fit into the above criteria.

THE REORDER QUERY

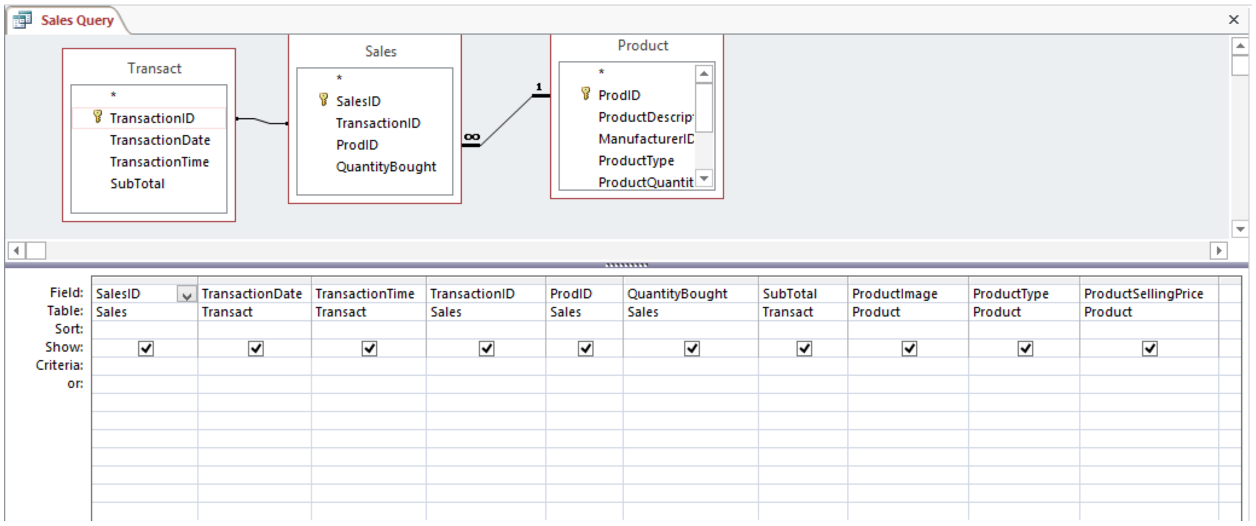
This query filters out the products that need reordering.



Field:	ProdID	ProductDescription	ManufacturerID	ProductQuantity	ProductSellingPrice	ProductReorderLevel		
Table:	Product	Product	Product	Product	Product	Product		
Sort:								
Show:	✓	✓	✓	✓	✓	✓	☐	☐
Criteria:				<=[ProductReorderLevel]		>=[ProductQuantity]		
or:								

THE SALES QUERY

Just combines the details of the transaction table, product table and the sales table to make one complete table.



Field:	SalesID	TransactionDate	TransactionTime	TransactionID	ProdID	QuantityBought	SubTotal	ProductImage	ProductType	ProductSellingPrice
Table:	Sales	Transact	Transact	Sales	Sales	Sales	Transact	Product	Product	Product
Sort:										
Show:	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Criteria:										
or:										