Blaire Pang

VBA Semester Project

MACRS Asset Database and Analysis

I. Executive Summary:

Many tax professionals face the challenge of keeping track of asset data, computing depreciation, and generating reports based on the data set. The manual tracking, calculation, and analysis process is mundane and prone to mistakes, especially when a taxpayer such as a corporation owns thousands of business assets. Instead of manually executing the tasks, I created a VBA-based solution where the user simply needs to click the customized ribbon buttons under the MACRS tab, then the spreadsheet will automatically, efficiently, and accurately accept new records, update the database, calculate depreciation for any asset in a particular year, and generate Pivot Tables based on information in the database.

II. Implementation Documentation:

Step 1: Setting up the Access Database.

The application depends entirely on the Access database. The application uses the database with the Asset, Type, Class, Convention, Method, A1, A2, A3, A4, A5, A6, and A7a tables. Asset, Type, Class, Convention, Method tables are related through TypeID, ClassID, ConventionID, and MethodID, as indicated by the relationship diagram in Figure 1.1.

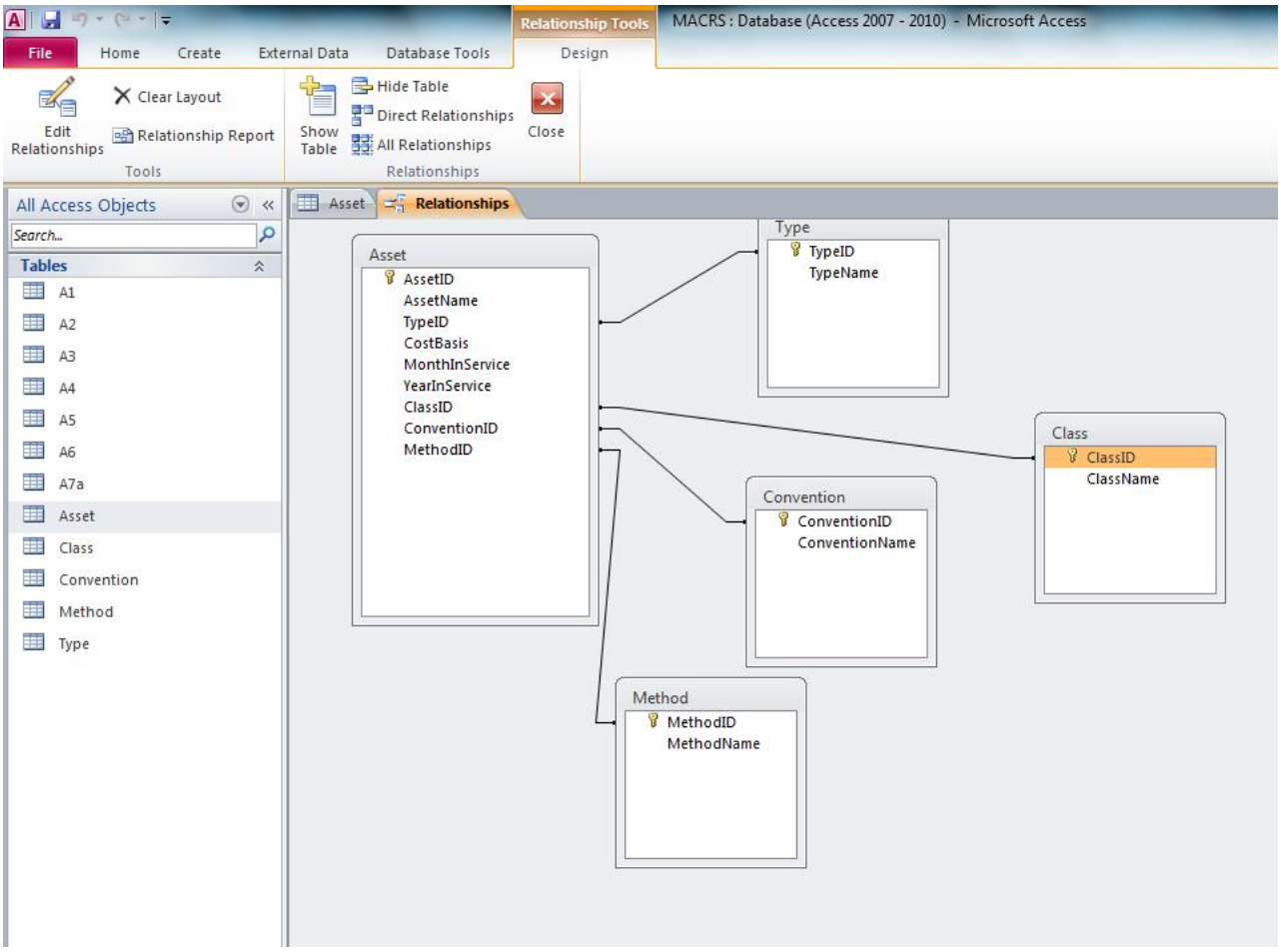


Figure 1.1

A1, A2, A3, A4, A5, A6, and A7a are tables of the MACRS depreciation rates. I downloaded the Excel files containing such data from the IRS Publication 946. Then I formatted the files and used the Import Wizard in Access to import the data as tables into the Access database.

Step 2. User Form "New Record"

The user form "New Record" allows the user to enter the information of a new asset, as shown in Figure 2.1.

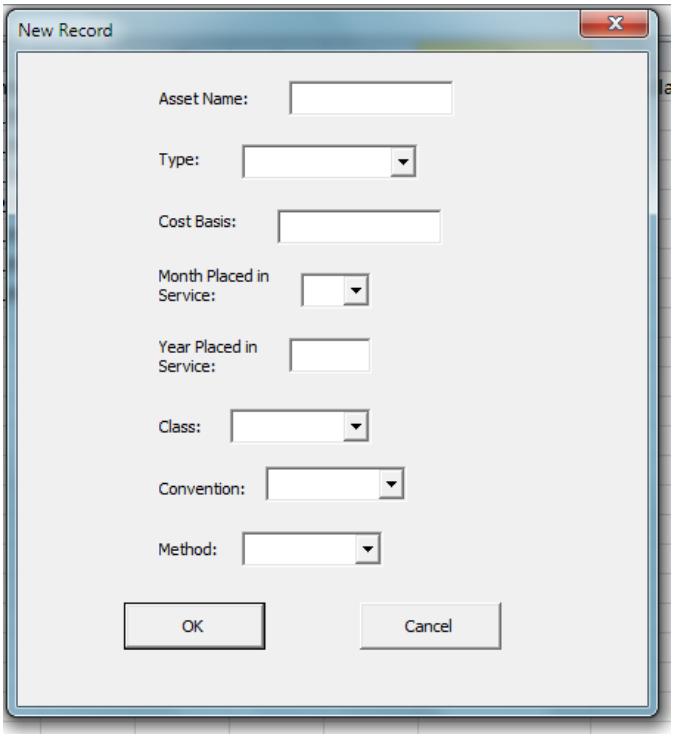


Figure 2.1

In creating the user form, I did the following:

1) I inserted a new user form in VBE and added the following controls into the user form: buttons, labels, text boxes, and combo boxes.
2) To get the controls to work properly, I edited the appropriate properties for the added controls.

After designing the user form, I opened the Code window and edited three event handlers: UserForm_Initialize, btnOK_Click, and btnCancel_Click. The first determines how the user form will look when the user first sees it, and the latter two determine what occurs when the user clicks on the OK and Cancel buttons.

1) UserForm_Initialize Code:

```
Private Sub UserForm_Initialize()
'Populate the boxes with the arrays
cboType.List = Types
cboClass.List = Classes
cboConv.List = Conventions
cboMeth.List = Methods
cboMonth.List = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
End Sub
```

The arrays are created based on the respected tables in the database. I created the arrays in Module 1 using SQL. Below is an example of how I created the Types() array.

```
Sub getTypeList()
    Dim SQL As String
    Dim nType As Integer

    SQL = "SELECT TypeID, TypeName FROM Type"
    With rs
        .Open SQL, cn
        nType = 0
        Do While Not .EOF
            nType = nType + 1
            ReDim Preserve Types(nType)
            Types(nType) = .Fields("TypeID") & ": " & .Fields("TypeName")
            Debug.Print Types(nType)
            .MoveNext
        Loop
        .Close
    End With
End Sub
```

2) btnOK_Click Code: this event handler captures the user's inputs. The inputs are stored in public variables: vName, vType,vBasis, vMonth, vYear, vClass, vConv, and vMeth. These variables are declared in Module 1, which pass to a sub procedure newRecord().

```
Private Sub btnOK_Click()
vName = assetName.text
vType = cboType.Value
vBasis = costBasis.text
vYear = yearS.text
vClass = cboClass.Value
vConv = cboConv.Value
vMeth = cboMeth.Value
vMonth = cboMonth.Value

Unload Me
End Sub
```

3) bthCancel_Click Code:

```
Private Sub btnCancel_Click()
    Unload Me
    End
```

The first line says to unload the user form (which is referred to with Me). This makes the form disappear. The second line says to end the program.

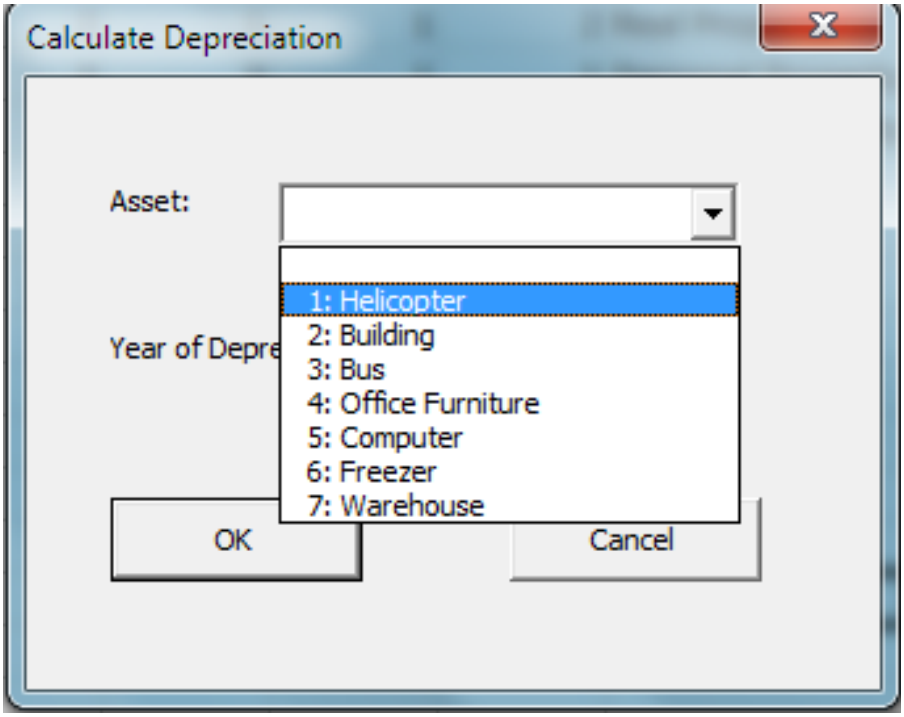
Step 3. Import the data in New Record into the Access Database

When the user clicks on the OK button in the New Record user form, the variables vName, vType, vBasis, vMonth, vYear, vClass, vConv, vMeth pass to the sub procedure newRecord() in Module 1. The sub Procedure opens the connection to the MACRS database, calls a couple of subs to create the arrays that populate the user form when it loads, receives the variables passed from the user form, and execute the SQL to insert the new record into the database.

```
Sub newRecord()
  Dim SQL As String

  Application.ScreenUpdating = False
  cn.Open "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & ThisWorkbook.Path & "\MACRS.accdb" & ";"

  Call getTypeList
  Call getClassList
  Call getConventionList
  Call getMethodList
  frm1.Show

  'Insert record
  SQL = "insert into Asset (AssetName, TypeID, CostBasis, MonthInService, YearInService, ClassID, ConventionID, metho
        "values( '" & vName & "', " & Split(vType, ": ")(0) & ", " & vBasis & ", " & vMonth & ", " & vYear & ", " & S
              Debug.Print SQL
  cn.Execute SQL

  cn.Close

  Application.ScreenUpdating = True

End Sub
```

Step 4. User Form "Calculate Depreciation"

The user form "Calculate Depreciation" allows the user to get the depreciation for any asset in a particular year. I used the same methods as in Step 2 to create and design the user form.



After designing the user form, I opened the Code window and edited three event handlers: UserForm_Initialize, btnOK_Click, and btnCancel_Click. The first determines how the user form will look when the user first sees it, and the latter two determine what occurs when the user clicks on the OK and Cancel buttons.

```
Private Sub bthCancel_Click()
    Unload Me
    End
End Sub

Private Sub bthOK_Click()
    wAsset = cboAsset.Value
    wYear = txtYear.text
    Unload Me
End Sub

Private Sub UserForm_Initialize()
    cboAsset.List = Assets
End Sub
```

After the user clicks on the OK button, the variables wAsset and wYear pass sub calculate() that does the calculation.

```
Application.ScreenUpdating = False
cn.Open "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & ThisWorkbook.Path & "\MACRS.accdb" & ";"

Call getAssetList
frm2.Show
```

The sub calculate() first opens the connection to the database, calls getAssetList to create arrays to populate the Asset combo box in the user form, then passes the input variables from the user form back into the SQL statement in the sub.

```
'Select record for requested asset
SQL = "SELECT * FROM Asset INNER JOIN Class on Asset.ClassID = Class.ClassID where AssetID = " & Split(wAsset, ": ")
Debug.Print SQL
Set rs = cn.Execute(SQL)
  cConv = rs.Fields("ConventionID").Value
  cYear = rs.Fields("YearInService").Value
  cMonth = rs.Fields("MonthInService").Value
  cClass = rs.Fields("ClassName").Value
  cBasis = rs.Fields("CostBasis").Value
  rs.Close

nYear = wYear - cYear + 1
nClass = Split(cClass, "-")(0)
```

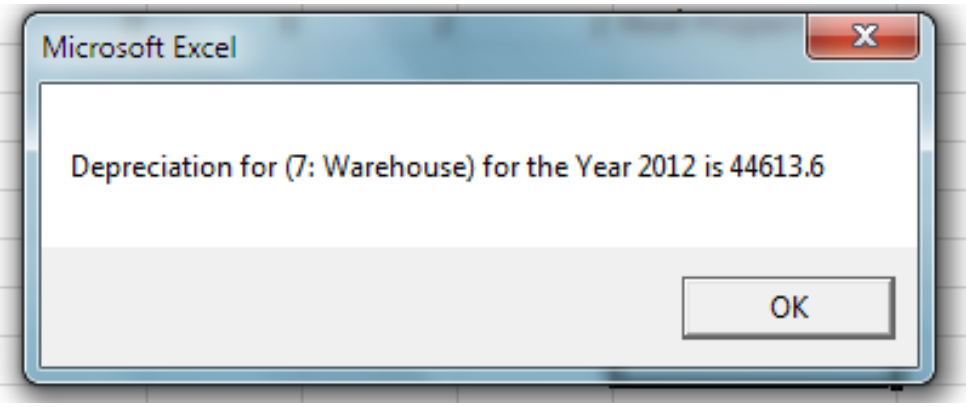
The SQL statement selects the entire record of the requested asset from the database. Then it captures the information necessary to determine which MACRS table to use to locate the appropriate rate.

Instead of writing six or seven IF statements to get to the right table, I concatenated the variable “cConv” into the table name which help locate the right rate. Samples of my codes are displayed as below.

```
  'Get Rate
'#Personal property
    If cConv < 6 Then
    SQL = "SELECT [" & nClass & "] from A" & cConv & " where Year = " & nYear
    Debug.Print SQL

    With rs
        .Open SQL, cn
        theRate = .Fields(0).Value
        Debug.Print theRate
        .Close
    End With

    End If
'#Real property
    If cConv = 6 Then
        If nClass = 27.5 Then
            SQL = "SELECT [" & cMonth & "] FROM A6 WHERE Year = " & nYear
            Debug.Print SQL
        End If

        If nClass = 39 Then
            SQL = "SELECT [" & cMonth & "] FROM A7a WHERE Year = " & nYear
            Debug.Print SQL
        End If

    With rs
        .Open SQL, cn
        theRate = .Fields(0).Value
        Debug.Print theRate
        .Close
    End With

    End If
```

After locating the appropriate rate, the following codes calculate the depreciation and display a message box with the depreciation figure to the user.

```
   'Calculate depreciation
     depre = theRate * cBasis
     Debug.Print depre
     MsgBox "Depreciation for (" & wAsset & ") for the Year " & wYear & " is " & depre
   cn.Close

   Application.ScreenUpdating = True

End Sub
```



Step 5. Export the Data from Access Database to Excel.

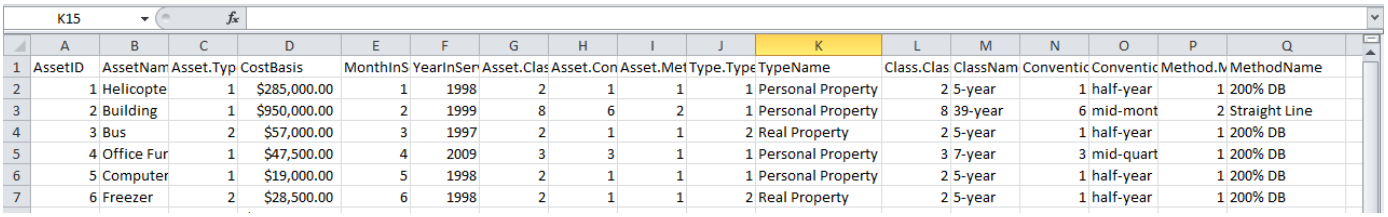
I created the sub procedure in the Ribbon module that contains the sub procedures for the Ribbon buttons. For this part, I used the ADO and SQL to join the tables and select all data into the spreadsheet. I also created a bar that shows on screen when the data is being imported. Samples of the codes are as below.

```
  Application.DisplayAlerts = False
  On Error Resume Next
    Sheets("Asset Data").Delete
  On Error GoTo 0
  Application.DisplayAlerts = True

  Set s = Worksheets.Add
  s.Name = "Asset Data"


  Set rs = cn.Execute("select count(*) from Asset")
  recordcount = rs.Fields(0).Value

  Set rs = cn.Execute("select * from ((((Asset INNER JOIN Type ON Asset.TypeID=Type.TypeID) INNER JOIN Class ON Asset.

  For c = 1 To rs.Fields.Count
    s.Cells(1, c).Value = rs.Fields(c - 1).Name
  Next
  r = 1
  frmProgress.Show False
  Do Until rs.EOF
    frmProgress.progress (r) / recordcount
    r = r + 1
    For c = 1 To rs.Fields.Count
      s.Cells(r, c).Value = rs.Fields(c - 1).Value
    Next
    rs.MoveNext
    DoEvents
  Loop
  Unload frmProgress

  rs.Close
  cn.Close

End Sub
```

```
Public Sub progress(percent As Single, Optional text As String)
  lblProgress.width = percent * 200
  lblPercent.Caption = Round(percent * 100, 1) & "%"
  If text > "" Then
     Me.height = 60
     lblText.Caption = text
  End If

End Sub
```

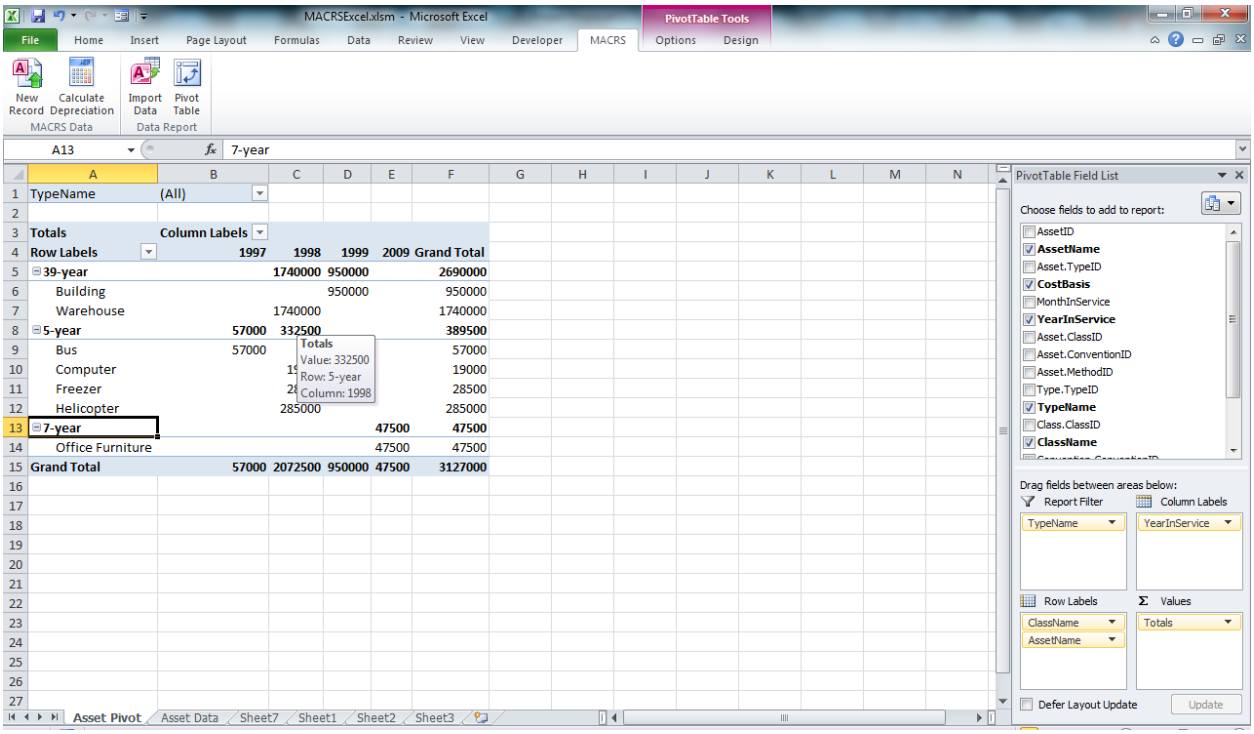
```
Private Sub UserForm_Initialize()
  lblProgress.width = 0
  lblPercent.Caption = "0%"
End Sub
```

Below is a screenshot of the data set after the import is complete.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	AssetID	AssetNam	Asset.Typ	CostBasis	MonthInS	YearInSer	Asset.Clas	Asset.Con	Asset.Met	Type.Type	TypeName	Class.Clas	ClassNam	Conventic	Conventic	Method.N	MethodName
2	1	Helicopte	1	$285,000.00	1	1998	2	1	1	1	Personal Property	2	5-year	1	half-year	1	200% DB
3	2	Building	1	$950,000.00	2	1999	8	6	2	1	Personal Property	8	39-year	6	mid-mont	2	Straight Line
4	3	Bus	2	$57,000.00	3	1997	2	1	1	2	Real Property	2	5-year	1	half-year	1	200% DB
5	4	Office Fur	1	$47,500.00	4	2009	3	3	1	1	Personal Property	3	7-year	3	mid-quart	1	200% DB
6	5	Computer	1	$19,000.00	5	1998	2	1	1	1	Personal Property	2	5-year	1	half-year	1	200% DB
7	6	Freezer	2	$28,500.00	6	1998	2	1	1	2	Real Property	2	5-year	1	half-year	1	200% DB

Step 6. Creating the Pivot Table based on the data set in Step 5.

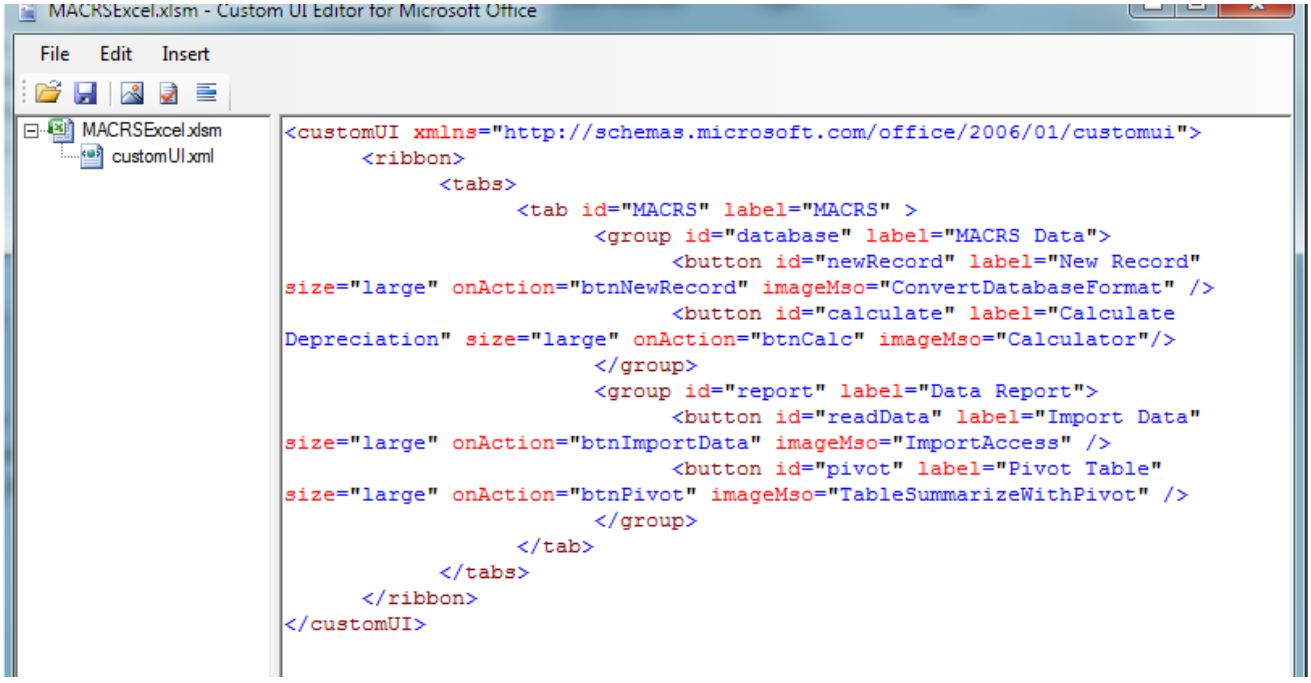


For this part, I created a sub procedure in the Ribbon Module that creates a Pivot Table based on the data set imported from Step 3. I first used Pivot Cache to store the data from Step 3, then created a Pivot Table to present such data. Samples of my code are shown as below.

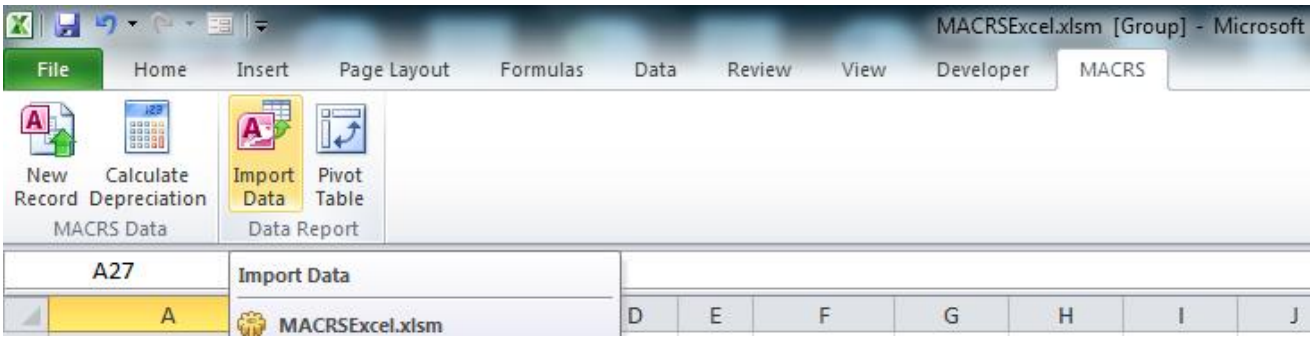
```
Sub btnPivot(control As IRibbonControl)
  Dim s As Worksheet
  Dim pc As PivotCache
  Dim pt As PivotTable
  Dim pf As PivotField

  Application.DisplayAlerts = False
  On Error Resume Next
    Sheets("Asset Pivot").Delete
  On Error GoTo 0
  Application.DisplayAlerts = True

  Set s = Worksheets.Add
  s.Name = "Asset Pivot"

  Set pc = ThisWorkbook.PivotCaches.Create(SourceType:=xlDatabase, SourceData:="'Asset Data'!" & Sheets("Asset Data")

  Set pt = pc.CreatePivotTable(TableDestination:="'Asset Pivot'!r1c1", TableName:="Asset", DefaultVersion:=xlPivotTab
  s.Select


    With pt.PivotFields("ClassName")
        .Orientation = xlRowField
        .Position = 1

    End With
    With pt.PivotFields("TypeName")
        .Orientation = xlPageField
        .Position = 1
    End With
```

Step 7. Creating the Ribbon "MACRS"

I used the Custom UI Editor to design the ribbon MACRS with buttons assigned with macros for the following functionality of my spreadsheet: New Record, Calculate Depreciation, Import Data, and Pivot Table.



After editing the code in the Custom UI Editor, I opened the excel file and edited the sub procedures in the Ribbon module so that the macros will run when suspected Ribbon buttons are clicked by the user. A screenshot of the ribbon is displayed as below.



III. Discussion of learning and conceptual difficulties encountered:

During the project, I have encountered the following difficulties:

1) Setting up the databases. With limited knowledge in Assess, I spent one hour to learn about the functionality of Access, and spent another two hours setting up the database. The most difficult part was formatting and importing the eight depreciation schedule tables. The original tables are in Word documents, with its data unorganized. I needed to transfer the eight sets of data into eight worksheets. After some manual formatting for half an hour, I figured out that I could write a VBA-coded macro to manipulate the data into an organized format. I subsequently wrote the code and successfully organized the data in the way that is easy to navigate and import as tables into the Access Database.
2) Writing SQL statements in VBA. I was not experienced using SQL in VBA. For this project, I needed to concatenate lots of strings and variables which involve ampersands, double quotes, single quotes, etc. It was easy to make mistakes and get confused in the process.

IV. Assistance:

Professor Allen has helped me debug a lot of errors and also taught me how to figure out writing the correct SQL statements in VBA.