

401(k) Pro, Inc. and James A. Gilbert Freeware End-User License Agreement

IMPORTANT - READ CAREFULLY

This 401(k) Pro, Inc. and James A. Gilbert Freeware End-User License Agreement ("EULA") is a legal agreement between you (either an individual person or a single legal entity, who will be referred to in this EULA as "User") and 401(k) Pro, Inc. and James A. Gilbert for the 401(k) Pro, Inc. and James A. Gilbert freeware product that accompanies this EULA, or that is subsequently provided to the User pursuant to further order, including any source codes, associated media, printed materials and electronic documentation (collectively, the "Freeware Product"). By installing, copying, downloading, accessing or otherwise using the Freeware Product, User agrees to be bound by the terms of this EULA. If User does not accept and agree to the terms and conditions of this EULA, 401(k) Pro, Inc. and James A. Gilbert is unwilling to license the Freeware Product contained herein to User. In that case do not install access or use the Freeware Product; instead delete the Freeware Product from your computer or network in whatever form(s) it exists.

MEANING OF WORDS USED IN THIS EULA

401(k) Pro, Inc. and James A. Gilbert: means 401(k) Pro, Inc. and James A. Gilbert and any or all of its divisions or subsidiaries.

User: the individual person or single legal entity that is deemed to be the licensed user of the Product under the terms of this EULA.

Freeware Product: means the 401(k) Pro, Inc. and James A. Gilbert Freeware Product that accompanies this EULA, including any source codes, associated media, printed materials and electronic documentation. The Freeware Product also includes any Freeware updates, Freeware version upgrades, Freeware configuration upgrades, add-on components, web services and/or supplements that 401(k) Pro, Inc. and James A. Gilbert may provide to User or make available to User after the date User obtains the initial copy of the Freeware Product, to the extent that such items are not accompanied by a separate license agreement or terms of use.

Freeware Bundle: refers to when one or more 401(k) Pro, Inc. and James A. Gilbert Freeware Products are supplied together as an inclusive package. When Freeware Products are supplied as a Freeware Bundle they are deemed a single Freeware Product in terms of their usage and transfer. In no event shall User be permitted to install any portion of a Freeware Bundle on a computer or network for which any of the Freeware Products are not licensed.

OWNERSHIP AND GRANT OF LICENSE

This EULA grants User permanent and non-exclusive right to: install and use the Freeware, and modify it at will.

FREEWARE PRODUCT TRANSFER

User may permanently transfer the entire Freeware Product to any third party without obtaining consent from 401(k) Pro, Inc. and James A. Gilbert

In the event that User has received this Freeware Product as part of a Freeware Bundle, then all Freeware Products supplied as part of that Freeware Bundle may be transferred. Freeware Product supplied as part of a Freeware Bundle be transferred separately.

WARRANTIES AND WARRANTY DISCLAIMERS; LIMITATION OF LIABILITY

THE FREEWARE PRODUCT IS PROVIDED ENTIRELY "AS IS", 401(k) Pro, Inc. and James A. Gilbert EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT TO THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ANY IMPLIED OR EXPRESS WARRANTIES OF TITLE

TO THE MAXIMUM EXTENT PERMITTED UNDER LAW, UNDER NO CIRCUMSTANCES WILL 401(k) Pro Inc. and James A. Gilbert BE LIABLE TO USER FOR ANY DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO ANY LOSS OF PROFITS, INTERRUPTION TO BUSINESS, LOSS OF INFORMATION OR ANY OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE SUPPLY TO OR USER'S USE OF OR INABILITY TO USE THE FREWARE PRODUCT. WHETHER 401(k) Pro, Inc. and James A. Gilbert HAS BEEN INFORMED OF THE POSSIBILITY OF THE SAME OR NOT, AND WHETHER STATED AS A CLAIM IN CONTRACT, TORT OR OTHER LEGAL THEORY.

USER ACKNOWLEDGES THAT NO PROMISE, REPRESENTATION, WARRANTY OR UNDERTAKING HAS BEEN MADE BY 401(k) Pro, Inc. and James A. Gilbert TO ANY PERSON OR COMPANY ON USER'S BEHALF AS TO THE PROFITABILITY OR ANY OTHER CONSEQUENCES OR BENEFITS TO BE OBTAINED FROM DELIVERY TO USER OF THE FREWARE PRODUCT. USER HAS RELIED UPON ITS OWN SKILL AND JUDGMENT IN ACQUIRING THE FREWARE PRODUCT. TO THE EXTENT APPLICABLE LAW DOES NOT PERMIT A COMPLETE LIMITATION OF DAMAGES AS SET FORTH HEREIN, USER AGREES THAT ITS DAMAGES SHALL BE TO ONE DOLLAR (U.S.) NOTHING HEREIN SHALL BE CONSTRUED AS ATTEMPTING TO ENFORCE RIGHTS AGAINST USER BEYOND THOSE PERMITTED BY APPLICABLE LAW.

IMPORTANT EXCEPTION TO LIMITATION OF LIABILITY WARRANTY

The above limitation may not apply to User as legislation, such as the Australian Trade Practices Act of 1974, may imply into this EULA a warranty or condition and such legislation may avoid or prohibit the terms of this EULA from excluding, restricting or modifying the application of such warranty or conditions. Nothing in this EULA is intended to exclude, restrict or modify the liability of 401(k) Pro, Inc. and James A. Gilbert under Part VA of the Australian Trade Practices Act of 1974 or any similar law under any other jurisdiction in which the Freeware Product is sold.

ENTIRE AGREEMENT

This EULA embodies the entire agreement between 401(k) Pro, Inc. and James A. Gilbert and User in relation to the terms on which User is licensed to use the Freeware Product; any written or oral agreement between the parties in connection with the subject matter hereof is hereby superseded and of no further force or effect. This EULA may be amended only by a writing executed by both parties.

GOVERNING LAW

This EULA is governed by the laws of the State of California, USA. 401(k) Pro, Inc. and James A. Gilbert and User each irrevocably and unconditionally submit to the exclusive jurisdiction and venue of the state and federal courts of California, USA, and each knowingly waives any objection that the courts therein do not have personal jurisdiction or that the courts are an improper forum, or not a convenient forum. The International Sale of Goods Convention shall not apply to this License Agreement, and is expressly excluded hereunder.

EXPORT RESTRICTIONS

Depending upon the jurisdiction in which the Freeware Product is sold, this License Agreement may be subject to certain government export and other restrictions. For example, if User utilizes Freeware Product in the United States, this License Agreement is subject to and conditional upon compliance with United States export regulations and User shall comply with all applicable laws and regulations of the United States, including export, relating to the Freeware Product and the use thereof, and User shall provide such documentation and assurances as are required from time to time to comply herewith.

SEVERANCE

If any provision of this EULA is or at any time becomes illegal, prohibited, void or unenforceable for any reason, that provision is severed from this EULA and the remaining provisions of this EULA shall continue to be enforceable and are to be construed with such additions, deletions and modifications as are necessary to give effect to the remaining provisions of this EULA.

SOURCE CODE

401keasy

```
' Module      : 401keasy
' Description: Precedures and functions used in this application
' Procedures  : EasyStartup
'             EasyStartup_2
'             fCheckValidPortfolio
'             fDecrypt
'             Value
'             fGetCompanyInfo
'             fGetCopyrightFooter
'             fGetDataPath
'             fGetEmployeeName
'             fSetTip
'             pDemoMessage
'             pExitTour
'             pSwitchToDemoData
'             pSwitchToRealData
'             sBeginMonthlyProcessing
'             sCheckAutoEnrollment

' Created   : Friday, August 13, 1999
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified   :
' Thursday, April 13, 2000 Cleaned with CodeTools
'
```

```
Option Compare Database
Option Explicit
```

```
Global gvDemo As Boolean
Global gvCompanyName As String
Global gvCompanyTaxID As String
Global gvCompanyAddress As String
Global gvCompanyCity As String
Global gvCompanyState As String
Global gvCompanyZip As String
Global gvCompanyPhone As String
Global gvDealerName As String
Global gvDealerAddress As String
Global gvDealerCity As String
Global gvDealerState As String
Global gvDealerZip As String
Global gvNASDRep As String
Global gvRepNumber As String
Global gvMaxEmployees As Integer
Global gvPlanYear As Integer
```

```
Public Function EasyStartup()
```

```
' Comments   : Function to display flash screen and license agreement
' Parameters : -
' Returns    : -
' Created    :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified   :
' Alison J. Balter to bring up splash without demo_startup
'
```

```
'Open form About401kEasy
```

```
DoCmd.OpenForm "About401kEasy", WindowMode:=acDialog 'Modified by Alison J. Balter to bring up splash WITHOUT startup
```

```

'Set timer for 5 seconds
Forms&[About401kEasy].TimerInterval = 5000
'Following lines were already commented out
'Do Until Forms.Count = 0
' DoEvents
'Exit

'Look in table EmployerData and check SuppressLicenseAgreement
If DLookup("SuppressLicenseAgreement", "EmployerData", "") = False Then
' SuppressLicenseAgreement is false so open form LicenseAgreement
DoCmd.OpenForm "LicenseAgreement"
Else
' SuppressLicenseAgreement is true so execute EasyStartup_2
EasyStartup_2
End If
End Function
Public Sub EasyStartup_2()
' Comments : Determine if demo version to run Deno_Startup or licensed version to get into the app
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

'5/19/2000 added RAH
Dim vReturnValue As Variant
Dim vCurrentPeriod As Single
Dim vLastPeriod As Single

'Check to see if demo version
If gvDemo Then
'Open from Demo_Startup
DoCmd.OpenForm "Demo_Startup"
'Exit procedure
Exit Sub
End If

'Check to see if there is a password
If IsNull(DLookup("SystemPassword", "DataGeneral", "")) Then
' Everything in this section must also be placed in the EnterPasscode screen
'Checks and moves eligibility criteria for all employees for EmployerID = 1
pCheckEligibilityAll 1
'Checks and moves vesting information for all employees for EmployerID = 1 and today's date
pCalculateVesting 1, Date
'Open form menu_Main
DoCmd.OpenForm "Menu_Main"

'Process to see if any employee has turned 70 and has not been notified as of today's date
vReturnValue = fOver70Check()
'Determine whether to display renewal message
vCurrentPeriod = datetoperiod(Date)
vLastPeriod = DLookup("LastExpirationReminder", "EmployerData", "")
If ((Year(Date) = gvPlanYear) And (Month(Date) <= 12) And (vCurrentPeriod < vLastPeriod)) Then DoCmd.OpenForm "RenewalOrderForm"
Else
'Open form EnterPasscode
DoCmd.OpenForm "EnterPasscode"
End If
End Sub

Public Function fCheckValidPortfolio(vPortfolioID)
' Comments : Insure the PortrfolioID is still on file
' Parameters : vPortfolioID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

'Check there is an actual value in PortfolioID
If IsNull(vPortfolioID) Then
    'Return false that passed PortfolioID is not on file
    fCheckValidPortfolio = False
    'Exit function
    Exit Function
End If

Dim db As Database
Dim rs As Recordset
Set db = DBEngine.Workspaces(0).Databases(0)
Set rs = db.OpenRecordset("Portfolios", dbOpenSnapshot)
'Look in table Portfolios for a match with the passed PortfolioID
rs.FindFirst "PortfolioID = " & vPortfolioID

'Check if match was found
If rs.NoMatch Then
    'No match found
    'Close recordset
    rs.Close
    'Return false that passed PortfolioID is not on file
    fCheckValidPortfolio = False
    'Exit function
    Exit Function
End If

'Match was found
'Check SuppressPortfolio
If rs.&SuppressPortfolio = True Then
    'Close recordset
    rs.Close
    'Return false that passed PortfolioID is not on file
    fCheckValidPortfolio = False
    'Exit function
    Exit Function
End If
'Close recordset
rs.Close
'return value true that passed PortfolioID is valid
fCheckValidPortfolio = True

End Function

Public Function fDecrypt(ByVal vString)
    ' Comments : Function to Decrypt a string value using a key that modifies each
    '            string in the string including two numbers that determine the length of
    '            padding for the front and back of the string. The encrypted string looks like:
    '            position
    '            1 key (random number 10 - 225) to modify every string that is encrypted
    '            2 front garbage number (random number 2 - 6)
    '            3 back garbage number (random number 2 - 6)
    '            4 - v the number of positions that equal the value of front garbage (padding)
    '            w - x the actual length of the passed string to encrypt
    '            y - z the number of positions that equal the value of back garbage (padding)
    '
    '            Using the values in the first three positions this procedure can then determine which positions
    '            hold the target data and what value to use to extract the actual data.
    ' Parameters : vString -
    ' Returns : -
    ' Created :
    ' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
    ' Modified :
    '
    ' _____

    'Enable error handling
    On Error GoTo Err_fDecrypt

    Dim vCount

```

```

Dim vKey
Dim vFrontGarbage
Dim vBackGarbage
Dim vCharAscii As Integer
Dim vNewString As String

'Check the passed string for a null or blank value
If IsNull(vString) Or (vString = "") Then
    'Value is null or blank
    'Set the return value to the value of the passed string
    fDecrypt = ""
    'Exit function
    Exit Function
End If

'Determine the value of the key stored in the first position of the field
vKey = Asc(Left$(vString, 1))

'Determine the value of vFrontGarbage stored in the second position of the field
'Take the ascii value of the field and subtract the value of vKey from it
vFrontGarbage = Asc(&(vString, 2, 1)) - vKey
'If the value is less than 0 then add 256 to the value
If vFrontGarbage < 0 Then vFrontGarbage = vFrontGarbage + 256
'Convert the value of the string to an integer and store it in vFrontGarbage
vFrontGarbage = CInt(Chr$(vFrontGarbage))

'Determine the value of vBackGarbage stored in the third position of the field
'Take the ascii value of the field and subtract the value of vKey from it
vBackGarbage = Asc(&(vString, 3, 1)) - vKey
'If the value is less than 0 then add 256 to the value
If vBackGarbage < 0 Then vBackGarbage = vBackGarbage + 256
'Convert the value of the string to an integer and store it in vBackGarbage
vBackGarbage = CInt(Chr$(vBackGarbage))

'Determine the actual value of the target string by stripping off the first three positions which carry
'the three key values and the extra positions for the length of the front garbage and back garbage
'Strip off the front garbage
vString = Right$(vString, Len(vString) - (3 + vFrontGarbage))
'Strip off the back garbage
vString = Left$(vString, Len(vString) - vBackGarbage)

'Set the new work area to blank
vNewString = ""
'Determine the true value for each position in the string
For vCount = 1 To Len(vString)
    'take the ascii value pf the string and subtract the value of the key modifier
    vCharAscii = Asc(&(vString, vCount, 1)) - vKey
    'If the result is negative and 256 to the number
    If vCharAscii < 0 Then vCharAscii = vCharAscii + 256
    'Convert the position to a string and append it to the new string
    vNewString = vNewString + Chr$(vCharAscii)
'Continue the next count
Next vCount

'Return the decrypted value of the string
fDecrypt = vNewString

'Exit the function
Exit Function

Err_fDecrypt:
'Display error message that there is decryption error to user
MsgBox "Critical Decryption Error. Application will exit.", vbCritical, PRODUCTNAME
'Exit function
Exit Function

End Function

Public Function Value(ByVal vString)
    ' Comments : Function to Encrypt a string value using a random number for a key that modifies each

```

```

' string in the string including two random numbers that determine the length of
' padding for the front and back of the string. The new encrypted string looks like:
' position
' 1 key (random number 10 - 225) to modify every string that is encrypted
' 2 front garbage number (random number 2 - 6)
' 3 back garbage number (random number 2 - 6)
' 4 - v the number of positions that equal the value of front garbage (padding)
' w - x the actual length of the passed string to encrypt
' y - z the number of positions that equal the value of back garbage (padding)
'
' The Decrypt procedure uses the values in the first three positions to extract the real string
' Parameters : vString -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

Dim vCount
Dim vLength
Dim vNewLength
Dim vKey
Dim vFrontGarbage
Dim vBackGarbage
Dim =Value() As Integer
Dim vNewString As String
Dim vStartOver As Boolean

```

```

'Check the passed string for a null or blank value
If IsNull(vString) Or (vString = "") Then
'Value is null or balnk
'Set the return value to the value of the passed string
Value = vString
'Exit function
Exit Function
End If

```

Value_Start:

```

'Set value to false
vStartOver = False

'Initialize the random-number generator
Randomize
'Get random number between 10 and 225 in vKey
vKey = Int((225 - 10 + 1) * Rnd + 10)
'Store random number between 2 and 6 in vFrontGarbage
vFrontGarbage = Int((6 - 2 + 1) * Rnd + 2)
'Store random number between 2 and 6 in vBackGarbage
vBackGarbage = Int((6 - 2 + 1) * Rnd + 2)

'Save the length of the passed string
vLength = Len(vString)
'Save the new lentgh that increases the old length by 3 positions that hold a key modifier and
'the value of two random numbers that will be used to fill the front and back of the string that
'is being encrypted
vNewLength = 3 + vFrontGarbage + vLength + vBackGarbage
'Redimension the size of the array to the value of the new length just calculated
ReDim =Value(1 To vNewLength) As Integer

'Save the three random numbers just created in the first three positions of the array
'Save the first random number used as a constant
=Value(1) = vKey
'Save the numerical value of vFrontGarbage converted to ascii value plus the value of vKey
=Value(2) = Asc(CStr(vFrontGarbage)) + vKey
'Save the numerical value of vBackGarbage converted to ascii value plus the value of vKey
=Value(3) = Asc(CStr(vBackGarbage)) + vKey

'Fill up the next few positions that equal the value of vFrontGarbage with random numbers between

```

```

'0 and 255
For vCount = 4 To (4 + vFrontGarbage - 1)
    'Generate the random number that will fill the next available position in the array
    =Value(vCount) = Int((255 - 0 + 1) * Rnd + 0)
    'Continue the next count
Next vCount

'Put the actual string that was passed into the array one string at a time after
'it has been modified by the vKey value
For vCount = 1 To vLength
    'One string at a time use the ascii value and add the value of vKey and store in the
    'next available position in the array
    =Value(vCount + 3 + vFrontGarbage) = Asc(&(vString, vCount, 1)) + vKey
    'Continue the next count
Next vCount

'Fill up the next few positions that equal the value of vBackGarbage with random numbers between
'0 and 255
For vCount = 1 To vBackGarbage
    'Generate the random number that will fill the next available position in the array
    =Value(vCount + 3 + vFrontGarbage + vLength) = Int((255 - 0 + 1) * Rnd + 0)
    'Continue the next count
Next vCount

'Check each position of the =Value for a numerical value
For vCount = 1 To vNewLength
    'If the value of the current position in =Value is greater than 255 then subtract 256 from it
    If =Value(vCount) < 255 Then =Value(vCount) = =Value(vCount) - 256
    'If the value of the current position in =Value is equal to 8, 9, 10, or 13 then set a to recalculate again
    If (=Value(vCount) = 8) Or (=Value(vCount) = 9) Or (=Value(vCount) = 10) Or (=Value(vCount) = 13) Then
        ' This tests for <BS<, <TAB<, <LF<, or <CR<
        'set to start over and recalculate
        vStartOver = True
        'Set the value of vCount to the length of the string so this process will stop
        vCount = vNewLength
    End If
'Continue the next count
Next vCount

'If this is true then this process must be rerun
If vStartOver = True Then GoTo Value_Start

'Set the value of vNewString to nothing
vNewString = ""
'For each position in =Value change to string using the current numeric value
For vCount = 1 To vNewLength
    'Change the value of this position in =Value to a string
    vNewString = vNewString & Chr$(=Value(vCount))
'Continue the next count
Next vCount

'Return the newly encrypted value
Value = vNewString

```

End Function

Public Function fGetCompanyInfo(vInfoType)

```

' Comments : Return a line of text of company information based on the passed parameter
' Parameters : vInfoType -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
'-----
'Evaluate the parameter
Select Case vInfoType
    'Return Company name
    Case 1
        fGetCompanyInfo = gvCompanyName

```

```
'Return Company address
```

```
Case 2
```

```
    fGetCompanyInfo = gvCompanyAddress
```

```
'Return Company Company city, state, and zip
```

```
Case 3
```

```
    fGetCompanyInfo = gvCompanyCity & ", " & gvCompanyState & " " & gvCompanyZip
```

```
'Return Company phone
```

```
Case 4
```

```
    fGetCompanyInfo = gvCompanyPhone
```

```
'Return Company tax ID
```

```
Case 5
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
```

```
'Find the record in MenuTps that matches the selected TipID
```

```
Set rs = db.OpenRecordset("Select Distinctrow * From MenuTips Where TipID = " & vTipID & ";", dbOpenSnapshot)
```

```
'Return the menu tip text to the calling form
```

```
Forms(vFormName)("TipText") = rs&TipText
```

```
'Close the recordset
```

```
rs.Close
```

```
End Function
```

```
Public Sub pDemoMessage()
```

```
' Comments : Function to display message to user explaining when user has tried to perform an action the
```

```
' demo version does not allow
```

```
' Parameters : -
```

```
' Returns : -
```

```
' Created :
```

```
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
```

```
' Modified :
```

```
'
```

```
'
```

```
' Display message to user about trial version of this product
```

```
MsgBox "You cannot perform this function in the trial version of " & PRODUCTNAME & ".", vbInformation, PRODUCTNAME & " Trial Software"
```

```
End Sub
```

```
Public Sub pExitTour()
```

```
' Comments : Close any open forms and reports. For demo version open form "Demo_Startup" or for licensed
```

```
' version open form "Menu_Uilities"
```

```
' Parameters : -
```

```
' Returns : -
```

```
' Created :
```

```
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
```

```
' Modified :
```

```
'
```

```
'
```

```
Dim vNumForms
```

```
Dim vNumReports
```

```
Dim i As Integer
```

```
Dim vFormName
```

```
Dim vReportName As String
```

```
'Save number of open forms
```

```
vNumForms = Forms.Count
```

```
'Start counter at zero and Exit
```

```
For i = (vNumForms - 1) To 0 Step -1
```

```
    'Save name of form referenced in this collection
```

```
    vFormName = Forms(i).Name
```

```
    'Close form name
```

```
    DoCmd.Close acForm, vFormName
```

```
Next i
```

```
'Save number of reports open
```

```
vNumReports = Reports.Count
```

```

        'Start counter at zero and Exit
For i = (vNumReports - 1) To 0 Step -1
    'Save name of report referenced in this collection
    vReportName = Reports(i).Name
    'Close report name
    DoCmd.Close acReport, vReportName
'Next count
Next i

'Check if demo version
If gvDemo = True Then
    'Is demo version
    'Open form Demo_Startup
    DoCmd.OpenForm "Demo_Startup"
Else
    'Is Not demo version
    'Get real data for this client
    pSwitchToRealData
    'Open form menu_Uilities
    DoCmd.OpenForm "Menu_Uilities"
End If
End Sub

Public Sub pSwitchToDemoData()
    ' Comments : Function to save client data to a holding file and switch in demo data for the guided tour
    ' Parameters : -
    ' Returns : -
    ' Created :
    ' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
    ' Modified :
    '
    '-----
    Dim vFilePath As String
    Dim vFileName1 As String
    Dim vFileName2 As String

    'Save the current path for the data currently in use
    vFilePath = fGetDataPath

    'Check to see if client has DemoData checked true in DataGeneral table
    If DLookup("DemoData", "DataGeneral", "") = False Then
        'DemoData is false
        'Save the current path and the standard file name the app uses
        vFileName1 = vFilePath & "401kdata.mdb"
        'Save the current path and a descriptive name to hold the client's real data file
        vFileName2 = vFilePath & "401kdata_real.mdb"
        'Copy the real data into the holding file
        FileCopy vFileName1, vFileName2
        vFileName1 = vFilePath & "401klic.mdb"
        vFileName2 = vFilePath & "401klic_real.mdb"
        FileCopy vFileName1, vFileName2
    End If

    'Save the current path and the file name that holds the demo data
    vFileName1 = vFilePath & "401kdata_demo.mdb"
    'Save the current path and the standard file name the app uses
    vFileName2 = vFilePath & "401kdata.mdb"
    'Copy the demo data into the standard app file name
    FileCopy vFileName1, vFileName2
    vFileName1 = vFilePath & "401klic_demo.mdb"
    vFileName2 = vFilePath & "401klic.mdb"
    FileCopy vFileName1, vFileName2
End Sub

Public Sub pSwitchToRealData()
    ' Comments : Function to switch the real data back after the guided tour has run
    ' Parameters : -
    ' Returns : -
    ' Created :
    ' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

```

```

' Modified :
'
'
'-----
Dim vFilePath As String
Dim vFileName1 As String
Dim vFileName2 As String

'Save the current path for the data currently in use
vFilePath = fGetDataPath

'Save the current path and a descriptive name that holds the client's real data file
vFileName1 = vFilePath & "401kdata_real.mdb"
'Save the current path and the standard file name the app uses
vFileName2 = vFilePath & "401kdata.mdb"
'Copy the real data back into the standard app file name
FileCopy vFileName1, vFileName2
vFileName1 = vFilePath & "401klic_real.mdb"
vFileName2 = vFilePath & "401klic.mdb"
FileCopy vFileName1, vFileName2
End Sub

Public Sub sBeginMonthlyProcessing(vProcessPeriod, vPostingPeriod)
' Comments : Prepare to process contributions for this processing period and posting period
' Parameters : vProcessPeriod -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
'-----
'Run procedure to create initial records for all employees for this process period and posting period
CreateContribData vProcessPeriod, vPostingPeriod
'Open form Contribution
DoCmd.OpenForm "Contribution"
'Set the value of the control Period to process period
Forms&[Contribution]&[Period] = vProcessPeriod
'Set the value of the control EmployerID to 1
Forms&[Contribution]&[EmployerID] = 1
'Set the record source to records equal to the process period
Forms&[Contribution]&[Detail].Form.RecordSource = "Select Distinctrow * from ContributionData Where [Period] = " & vProcessPeriod & "
Order by Name;"
'Close the form ContributionSetup
DoCmd.Close acForm, "ContributionSetup"
'Set focus on the form Contribution
Forms&[Contribution].SetFocus
End Sub

Public Sub sCheckAutoEnrollment(vProcessPeriod, vPostingPeriod)
' Comments : Check criteria to see if Auto enrollment is used or not and depending on result open
' the auto enrollment form or start processing records
' Parameters : vProcessPeriod -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
'-----
'Check if any employees are eligible, not a participant, and suppress auto enrollment is false
If IsNull(DLookup("EmployeeID", "EmployeeData", "Eligible=True AND Participant=False AND SuppressAutoEnrollment=False")) Then
'No employees are found that need auto enrollment so process the records for the selected process period and posting period
sBeginMonthlyProcessing vProcessPeriod, vPostingPeriod
Else
'There are employees that need to be enrolled
'Open form AutoEnrollment
DoCmd.OpenForm "AutoEnrollment"
'Set the value of the control processperiod to process period
Forms&[AutoEnrollment]&[ProcessPeriod] = vProcessPeriod
'Set the value of the control PostingPeriod to posting period
Forms&[AutoEnrollment]&[PostingPeriod] = vPostingPeriod

```

```
        'Set the record source to equal all employees tah are eligible, not a participant, and suppress auto enrollment is false
        Forms.&[AutoEnrollment]&[Detail].Form.RecordSource = "Select Distinctrow * From EmployeeData Where Eligible = True AND Participant =
False AND SuppressAutoEnrollment = False Order By [Last Name], [First Name];"
    End If
End Sub
```

401(K)Pro

```
' Module   : 401(K) Pro
' Description: Functions and procedures specific to 401keasy
' Procedures : ConvertPeriodToString
'
'   datetoperiod
'   fCheckEligibility
'   fGetAge
'   fGetCurrentLoanPayoff
'   fGetDemo
'   fGetITDLiquidations
'   fGetITDPurchases
'   fGetMonths
'   fGetPlanType
'   fOver70Check
'   fSaveRecord
'   GetITDAdjustments
'   GetITDContributions
'   GetITDMatching
'   GetITDRollovers
'   GetITDTransfers
'   GetLastPeriod
'   GetLastPeriodForYear
'   GetLicense
'   GetNewEmployeeID
'   GetPassCodeText
'   GetTotContrib
'   GreaterOf
'   InitVars
'   LesserOf
'   pCalculateTestingResults
'   pCalculateVesting
'   pCheckEligibilityAll
'   pClearTempADPData
'   pClearTempBatchData
'   pClearTempVestingData
'   pCreateMFApp
'   pCreateTempADPData
'   pCreateTempBatchData
'   pCreateTempVestingData
'   pCreateTempYTDActivityByFamily
'   pCreateTempYTDActivityByPortfolio
'   pCreateYTDSalaryGrid
'   periodtodate
'   perminus
'   perplus
'   pLoadTempBatchData
'   pSaveRecord
'   SetEstimatedContrib
'   SetEstimatedMatch
'   SetEstimatedSalary
'   SetHCandKey
'   SetHCandKeyADP
'   SetPriorYearSalary
'   SetYTDCompensation

' Created   : Friday, August 13, 1999
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified   :
' Thursday, April 13, 2000 Cleaned with CodeTools

Option Compare Database 'Use database order for string comparisons
Option Explicit
```

Global adjustmentidnumber As Long 'Used to post adjustments to AllocatedFunds
 Global drive As String 'Used to set the disk drive
 Global curperiod As Single 'Set in PrintMonthlyReports procedure
 Global STATISTICSYEAR As Integer 'Used for Contribution Statistics Report
 Global MFAppEmpSS As String 'Not referenced in application
 Global PRINTOUTPUT As Integer 'Determines printer or screen output
 Global gEmployerID As Long 'Not referenced in application
 Global gYear As Integer 'Not referenced in application
 Global gReportLog As Integer 'Used to see if report has been logged before printing

Global g5500_Year As Integer

Function ConvertPeriodToString(Period)

' Comments : Convert the period format from number like 1999.01 to string Jan 1999 using abbreviated month and year

' Parameters : Period -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

'

' _____

' Converts the decimal period to a string value.

Dim db As Database

Dim rs As Recordset

Dim periodyear

Dim periodmonth As Integer

'5/22/2000 added RAH

Dim monthstring As String

Dim yearstring As String

Set db = DBEngine.Workspaces(0).Databases(0)

'Check period to see if it has a value

If Period << 0 Then

'Save yyyy part of the period number

periodyear = Int(Period)

'Save the mm part of the period number by subtracting the whole number year and multiplying the decimal year

'by 100 to get a whole number

periodmonth = (Period - periodyear) * 100

'Read the record that corresponds to the month number

Set rs = db.OpenRecordset("Select Distinctrow * From T_Months Where MonthID = " & periodmonth & ":", dbOpenSnapshot)

'Save the abbreviated month for this month number

monthstring = rs&MonthAbbr

'Close the recordset

rs.Close

'Convert the year number to a string and save

yearstring = CStr(periodyear)

'Set return value to the abbreviated month a blank and the year

ConvertPeriodToString = monthstring & " " & yearstring

Else

'Period was zero

'Set return value to None

ConvertPeriodToString = "None"

End If

End Function

Function datetoperiod(dateval)

' Comments : Convert a date to the decimal format yyyy.mm

' Parameters : dateval -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

'

' _____

' Converts a date to a decimal period value.

'Return selected date by adding the year to the month divided by 100 and rounded to two places
datetoperiod = cfix2(Year(dateval) + (Month(dateval) / 100))

End Function

Public Function fCheckEligibility(vEmployerID, vEmployeeID)

' Comments : Return true if employee is eligible for plan based on employer criteria

' Parameters : vEmployerID -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

'

Dim db As Database

Dim rs As Recordset

Dim vEligibilityAge As Integer

Dim vEligibilityMonths As Integer

Dim vEligibilityPartTimers As Boolean

Dim vEligibilityUnionMembers As Boolean

Dim vEligibilityNonResidentAliens As Boolean

Dim vBirthDate As Date

Dim vHireDate As Date

Dim vTerminated As Boolean

Dim vPartTimeEmployee As Boolean

Dim vUnionMember As Boolean

Dim vNonResidentAlien As Boolean

Set db = DBEngine.Workspaces(0).Databases(0)

'Get the employer record for the selected employer

Set rs = db.OpenRecordset("Select Distinctrow * From EmployerData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)

'If either the employee's eligibility age is null or the eligibility months is null return false

If IsNull(rs&EligibilityAge) Or IsNull(rs&EligibilityMonths) Then

'Close the recordset

rs.Close

'Return false

fCheckEligibility = False

'Exit function

Exit Function

End If

'Save data from the EmployerData table and close

'Save the eligibility age

vEligibilityAge = rs&EligibilityAge

'Save the eligibility months

vEligibilityMonths = rs&EligibilityMonths

'Save the eligible part timer

vEligibilityPartTimers = rs&EligibilityPartTimers

'Save the eligibility union members

vEligibilityUnionMembers = rs&EligibilityUnionMembers

'Save the eligibilty non resident members

vEligibilityNonResidentAliens = rs&EligibilityNonResidentAliens

'Close recordset

rs.Close

'Get the employee record for the selected employee

Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData Where EmployeeID = " & vEmployeeID & ";", dbOpenSnapshot)

'Save the value of terminated

If rs&[Terminated] = "Terminated" Then vTerminated = True Else vTerminated = False

'Save the value of partTimeEmployee

vPartTimeEmployee = rs&PartTimeEmployee

'Save the value of UnionMember

vUnionMember = rs&UnionMember

'Save the value of NonresidentAlien

vNonResidentAlien = rs&NonResidentAlien

'If the employee's birthdate is null or hire date is null return false

If IsNull(rs&BirthDate) Or IsNull(rs&[Hire Date]) Then

```

    'Close recordset
rs.Close
    'Return false
fCheckEligibility = False
Else
    'Save the birth date
vBirthDate = rs&BirthDate
    'Save the hire date
vHireDate = rs&[Hire Date]
    'Close recordset
rs.Close

    'If all criteria is not met then employee is not eligible
    'Employee must not be terminated
    If vTerminated = True Then
        'Return false
        fCheckEligibility = False
    'If union members are not eligible employee must not be union member
    ElseIf vEligibilityUnionMembers = False And vUnionMember = True Then
        'Return false
        fCheckEligibility = False
    'If non resident aliens are not eligible employee must not be non resident alien
    ElseIf vEligibilityNonResidentAliens = False And vNonResidentAlien = True Then
        'Return false
        fCheckEligibility = False
    'If part timers are not eligible employee must not be part time
    ElseIf vEligibilityPartTimers = False And vPartTimeEmployee = True Then
        'Return false
        fCheckEligibility = False
    'Employee must be older than eligibility age
    ElseIf vEligibilityAge < fGetAge(vEmployeeID) Then
        'Return false
        fCheckEligibility = False
    'Employee must have worked more months than the eligibility months
    ElseIf vEligibilityMonths < fGetMonths(vHireDate, Date) Then
        'Return false
        fCheckEligibility = False
    Else
        'Employee meets all criteria
        'Return true
        fCheckEligibility = True
    End If
End If

```

End Function

Public Function fGetAge(vEmployeeID)

```

' Comments : Return the selected employee's age as of today as an integer of years
' Parameters : vEmployeeID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

Dim db As Database
Dim rs As Recordset
Dim vNumDays As Long

```

```

Set db = DBEngine.Workspaces(0).Databases(0)
'Get selected employee record
Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData Where EmployeeID = " & vEmployeeID & ";", dbOpenSnapshot)
'Return the age in years based on the birthday
If IsNull(rs&BirthDate) Then
    'Birth date is null so return zero
    fGetAge = 0
Else

```

```

    'Calculate the number of days from employee's burth date through today
    vNumDays = DateDiff("d", rs&BirthDate, Now) + 1
    'Divide the number of days old by average number of days in year (leap year adjusted) and return
    'the integer value
    fGetAge = Int(vNumDays / 365.25)
End If
'Close recordset
rs.Close

```

End Function

```

Public Function fGetCurrentLoanPayoff(vEmployerID, vEmployeeID)
' Comments : Calculate the amaount needed to pay off an employee loan for selected employee and employer
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Dim rs As Recordset
Set db = DBEngine.Workspaces(0).Databases(0)
'Get the selected employee record with a loan amount balance
Set rs = db.OpenRecordset("Select Distinctrow * From LoanInfo Where EmployerID = " & vEmployerID & " And EmployeeID = " & vEmployeeID
& " And LoanAmount < LoanPrincipalPaid;", dbOpenSnapshot)
'Check if record was returned
If rs.EOF Then
    'No record found so return N/A for balance
    fGetCurrentLoanPayoff = "N/A"
Else
    'Calculate the current loan balance equal to original loan amount less pricipal payments rounded to two places
    fGetCurrentLoanPayoff = cfix2(rs&LoanAmount - rs&LoanPrincipalPaid)
End If
'Close recordset
rs.Close
End Function

```

```

Function fGetDemo(vCompanyName, vSoftwareRegistration, vSoftwareKey)
' Comments : Determines if this is demo or licensed version based on criteria in the EmployerData table
' Parameters : vCompanyName
'             vSoftwareKey -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

Dim vDecryptedKey As Variant

```

```

'Decrypt the selected software key
vDecryptedKey = fDecrypt(vSoftwareKey)

```

```

'If there is no software key this is a demo

```

```

If vDecryptedKey = "" Then

```

```

    'Return true
    fGetDemo = True
    'Exit function
    Exit Function
End If

```

```

'If the decrypted key does not contain the same company name this is a demo

```

```

If Left$(vDecryptedKey, Len(vDecryptedKey) - 5) << vCompanyName Then

```

```

    'Return true
    fGetDemo = True
    'Exit function
    Exit Function
End If

```

```

End If

```

```

'If the decrypted key does not contain the same software registration this is a demo
If Right$(vDecryptedKey, 5) << Right$(vSoftwareRegistration, 5) Then
    'Return true
    fGetDemo = True
    'Exit function
    Exit Function
End If

```

```

'This is a licensed version for this company
'Return false
fGetDemo = False

```

End Function

Public Function fGetITDLiquidations(vEmployerID, vEmployeeID)

```

' Comments : Calculate the sum of TotAmount for selected employee and employer
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
Dim db As Database
Dim rs As Recordset
Set db = DBEngine.Workspaces(0).Databases(0)
'Sum all of the TotAmount for selected employee and employer from AllocatedFunds table
Set rs = db.OpenRecordset("Select Sum(Amount) As TotAmount From AllocatedFunds Where EmployerID = " & vEmployerID & " And
EmployeeID = " & vEmployeeID & " And Amount < 0;", dbOpenSnapshot)
'Return the total rounded to two places
fGetITDLiquidations = cfix2(nullto0(rs&TotAmount))
'Close recordset
rs.Close
End Function

```

Public Function fGetITDPurchases(vEmployerID, vEmployeeID)

```

' Comments : Calculate the sum of TotAmount for selected employee and employer
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
Dim db As Database
Dim rs As Recordset
Set db = DBEngine.Workspaces(0).Databases(0)
'Sum all of the TotAmount for selected employee and employer from AllocatedFunds table
Set rs = db.OpenRecordset("Select Sum(Amount) As TotAmount From AllocatedFunds Where EmployerID = " & vEmployerID & " And
EmployeeID = " & vEmployeeID & " And Amount < 0;", dbOpenSnapshot)
'Return the total rounded to two places
fGetITDPurchases = cfix2(nullto0(rs&TotAmount))
'Close recordset
rs.Close
End Function

```

Public Function fGetMonths(vDate1, vDate2)

```

' Comments : Calculate the number of months between two dates as an integer
' Parameters : vDate1 -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

Dim vNumDays As Long

```

'Determine the number of days between the two dates

```

```
vNumDays = DateDiff("d", vDate1, vDate2) + 1
'Return the number of months as days divided by the average number of days in a month as a whole number
fGetMonths = Int(vNumDays / 30.4375)
```

End Function

Public Function fGetPlanType(vEmployerID)

```
' Comments : Get the name of the plan type for the selected employer
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
Dim db As Database
Dim rs As Recordset
Dim vPlanTypeID As Long
Dim vPlanType As String
Set db = DBEngine.Workspaces(0).Databases(0)
'Get the employer record for the selected employer from the EmployerData table
Set rs = db.OpenRecordset("Select Distinctrow * From EmployerData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Save the PlanTypeID from the employer record
vPlanTypeID = rs&PlanTypeID
'Get the plan record from the PlanTypes table
Set rs = db.OpenRecordset("Select Distinctrow * From PlanTypes Where PlanTypeID = " & vPlanTypeID & ";", dbOpenSnapshot)
'Save the name of the plan type
vPlanType = rs&PlanType
'Close the
rs.Close
'Return plan type name
fGetPlanType = vPlanType
```

End Function

Public Function fOver70Check()

```
' Comments : Determine if any employees are over 70 as of today and have not been notified
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
Dim vDateVar As Date
'Save the date of 70 years ago today
vDateVar = Date - 25567
'Look in EmployeeData table for any employees who might have recently turned 70 and have not been notified
'Exit procedure if there are none
If IsNull(DLookup("EmployeeID", "EmployeeData", "Over70Notified = False and BirthDate <= #" & vDateVar & "#")) Then Exit Function
'If found any employees then open form Over70
DoCmd.OpenForm "Over70"
'Set the recordsource for the form as all employees over 70 who have not been notified
Forms&[Over70]&[Detail].Form.RecordSource = "Select Distinctrow * From Over70Q Where Over70Notified = False and BirthDate <= #" & vDateVar & "# Order By Company, EmployeeName;"
End Function
```

Public Function fSaveRecord()

```
' Comments : Generic function to save the current record
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
'Save the current record
```

```
DoCmd.RunCommand acCmdSaveRecord
End Function
```

```
Function GetITDAdjustments(employeridnum, employeeidnum)
```

```
‘ Comments : Calculate all adjustments for selected employer and employee
‘ Parameters : employeridnum -
‘ Returns : -
‘ Created :
‘ Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
‘ Modified :
‘
‘ _____
```

```
‘ Calculates total Inception To Date Adjustments for the Employee Monthly Statement.
```

```
Dim db As Database
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
‘Add all totadjustments from AllocatedFunds table for ContributionID = 0 and source is not 5 or 6 for selected employer and employee
Set rs = db.OpenRecordset(“select sum(Amount) as totadjustments from AllocatedFunds where ContributionID = 0 And Source << 5 And Source << 6 and EmployerID = “ & employeridnum & “ and EmployeeID = “ & employeeidnum & “;”, dbOpenSnapshot)
‘Return the total of adjustments rounded to two places
GetITDAdjustments = cfix2(nullto0(rs&totadjustments))
```

```
End Function
```

```
Function GetITDContributions(employeridnum, employeeidnum)
```

```
‘ Comments : Calculate all contributions for selected employer and employee
‘ Parameters : employeridnum -
‘ Returns : -
‘ Created :
‘ Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
‘ Modified :
‘
‘ _____
```

```
‘ Calculates total Inception To Date Contributions for a selected employee.
```

```
Dim db As Database
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
‘Add all contributions from AllocatedFunds table for selected employer and employee and source = 1
Set rs = db.OpenRecordset(“select sum(Amount) as totalcontributions from AllocatedFunds where EmployerID = “ & employeridnum & “ and EmployeeID = “ & employeeidnum & “ and Source = 1;”, dbOpenSnapshot)
‘Return the total of contributions rounded to two places
GetITDContributions = cfix2(nullto0(rs&TotalContributions))
```

```
End Function
```

```
Function GetITDMatching(employeridnum, employeeidnum)
```

```
‘ Comments : Calculate all matching contributions for selected employer and employee
‘ Parameters : employeridnum -
‘ Returns : -
‘ Created :
‘ Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
‘ Modified :
‘
‘ _____
```

```
‘ Calculates Inception To Date Employer Matches for selected Employee.
```

```
Dim db As Database
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
‘Add all matching contributions from AllocatedFunds table for selected employer and employee and source = 2
```

```
Set rs = db.OpenRecordset("select sum(Amount) as totmatch from AllocatedFunds where EmployerID = " & employeridnum & " and EmployeeID = " & employeeidnum & " and Source = 2;", dbOpenSnapshot)
'Return the total of matching contributions rounded to two places
GetITDMatching = cfix2(nullto0(rs&TotMatch))
```

End Function

Function GetITDRollovers(employeridnum, employeeidnum)

```
' Comments : Calculate all rollovers for selected employer and employee
' Parameters : employeridnum -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

' Calculates Inception To Date Rollovers for selected Employee.

```
Dim db As Database
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
'Add all rollovers from AllocatedFunds table for selected employer and employee and source = 5
Set rs = db.OpenRecordset("select sum(Amount) as totalrollovers from AllocatedFunds where Source = 5 and EmployerID = " & employeridnum & " and EmployeeID = " & employeeidnum & ";", dbOpenSnapshot)
'Return the total of rollovers rounded to two places
GetITDRollovers = cfix2(nullto0(rs&totalrollovers))
```

End Function

Function GetITDTransfers(employeridnum, employeeidnum)

```
' Comments : Calculate all transfers for selected employer and employee
' Parameters : employeridnum -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

' Calculates Inception To Date Trust Transfers for selected Employee.

```
Dim db As Database
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
'Add all transfers from AllocatedFunds table for selected employer and employee and source = 6
Set rs = db.OpenRecordset("select sum(Amount) as tottransfers from AllocatedFunds where Source = 6 and EmployerID = " & employeridnum & " and EmployeeID = " & employeeidnum & ";", dbOpenSnapshot)
'Return the total of transfers rounded to two places
GetITDTransfers = cfix2(nullto0(rs&TotTransfers))
```

End Function

Function GetLastPeriod(employeridnum)

```
' Comments : Returns a string value for the last processed period for a client in format
' Month day, year (January 31, 1999)
' Parameters : employeridnum -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

'5/22/2000 added RAH

```
Dim LastPeriod As String
```

' Returns a string value for the last processed period for a client.

'Get the last period from EmployerData for selected Employer

```
LastPeriod = DLookup("LastPeriod", "EmployerData", "EmployerID = " & employeridnum)
'Return the period in the converted format
GetLastPeriod = periodtodate>LastPeriod, 2)
```

End Function

Function GetLastPeriodForYear(vEmployerID, vYear)

```
' Comments : Calculates the last period that was processed for a selected client for a selected year.
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
'If either EmployerID or year is null cannot run this procedure
```

```
If IsNull(vEmployerID) Or IsNull(vYear) Then
```

```
'Set return value to 0
```

```
GetLastPeriodForYear = 0
```

```
'Exit procedure
```

```
Exit Function
```

```
End If
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
```

```
'Find the highest period processed for the selected EmployerID and year
```

```
Set rs = db.OpenRecordset("Select Max(Period) as maxperiod from ContributionData where EmployerID = " & vEmployerID & " and Int(Period) = " & vYear & ";", dbOpenSnapshot)
```

```
'Set return value to value of highest period
```

```
GetLastPeriodForYear = rs&maxperiod
```

```
'Close the recordset
```

```
rs.Close
```

End Function

Function GetLicense()

```
' Comments : Verifies that a valid license disk has been installed.
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
' Verifies that a valid license disk has been installed.
```

```
'Enable error handling
```

```
On Error GoTo ERR_GETLICENSE
```

```
Dim db As Database
```

```
Dim rs As Recordset
```

```
Dim vPlanYear As String
```

```
Dim vLicenseYear As String
```

```
Dim vReturnValue As Variant
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
```

```
'Open the EmployerData table
```

```
Set rs = db.OpenRecordset("EmployerData", dbOpenSnapshot)
```

```
'Read the License year in the first record
```

```
vLicenseYear = rs&LicenseYear
```

```
'Open the General table
```

```
Set rs = db.OpenRecordset("General", dbOpenSnapshot)
```

```
'Read the Plan year in the first record
```

```
vPlanYear = rs&PlanYear
```

```
'Close the recordset
```

```
rs.Close
```

```

'Determine if license year in EmployerData table is same as plan year in General table
'If vPlanYear = vLicenseYear Then
'  'Open form UpdateLicense
'  DoCmd.OpenForm "UpdateLicense"
'  'Exit function
'  Exit Function
'End If

```

```

'Determine if decrypted license year in EmployerData table is same as decrypted plan year in General table
If fDecrypt(vPlanYear) << fDecrypt(vLicenseYear) Then
  'Open form UpdateLicense
  DoCmd.OpenForm "UpdateLicense"
  'Save the company name to a global variable
  gvCompanyName = fDecrypt(rs&Company)
  'Save the company tax ID to a global variable
  gvCompanyTaxID = fDecrypt(rs&[Tax ID])
  'Save the company address to a global variable
  gvCompanyAddress = fDecrypt(rs&Address)
  'Save the company city to a global variable
  gvCompanyCity = fDecrypt(rs&City)
  'Save the company state to a global variable
  gvCompanyState = fDecrypt(rs&State)
  'Save the company zip to a global variable
  gvCompanyZip = fDecrypt(rs&Zip)
  'Save the company phone to a global variable
  gvCompanyPhone = fDecrypt(rs&Phone)
  'Save the dealer name to a global variable
  gvDealerName = fDecrypt(rs&DealerName)
  'Save the dealer address to a global variable
  gvDealerAddress = fDecrypt(rs&DealerHomeAddress)
  'Save the dealer city to a global variable
  gvDealerCity = fDecrypt(rs&DealerHomeCity)
  'Save the dealer store to a global variable
  gvDealerState = fDecrypt(rs&DealerHomeState)
  'Save the dealer zip to a global variable
  gvDealerZip = fDecrypt(rs&DealerHomeZip)
  'Save the NASD rep to a global variable
  gvNASDRep = fDecrypt(rs&NASDRep)
  'Save the rep number to a global variable
  gvRepNumber = fDecrypt(rs&[Rep Number])
  'Pass the company name, software registration and software key to determine if this is licensed version
  gvDemo = fGetDemo(gvCompanyName, rs&SoftwareRegistration, rs&SoftwareKey)
  'Save the maximum number of employees this license is for to a global variable
  gvMaxEmployees = CInt(fDecrypt(rs&MaxEmployees))
  'Save the plan year this license is for to a global variable
  gvPlanYear = CInt(fDecrypt(rs&LicenseYear))

```

```

rs.Close
Set rs = Nothing
Set db = Nothing

```

```

'If this is a licensed version and the DemoData field in DataGeneral table is true then copy real data into 401kdata.mdb
If Not gvDemo And DLookup("DemoData", "DataGeneral", "") Then pSwitchToRealData

```

```

'Start application
vReturnValue = EasyStartup()

```

End Function

Function LesserOf(number1, number2)

```

' Comments : Determine the lesser of two selected numbers
' Parameters : number1 -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

'Save the first number
number1 = nullto0(number1)

```

```

'Save the second number
number2 = nullto0(number2)
'If the first number is lesser return it or if the second number is lesser return it
If number1 < number2 Then LesserOf = number1 Else LesserOf = number2
End Function

```

```
Public Sub pCalculateTestingResults()
```

```

' Comments : Calculates testing results for highly compensated and key employees calculated on
' actual salary or estimated salary
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

Dim db As Database
Dim rs As Recordset
Dim vTotHCADP
Dim vTotNHCADP
Dim vTotHCACP
Dim vTotNHCACP As Single
Dim vTotKey
Dim vTotNonKey As Currency
Dim vNumHC
Dim vNumNHC As Integer
Set db = DBase.Workspaces(0).Databases(0)

```

```

'Set mouse to hourglass
DoCmd.Hourglass True

```

```

'Return the value of TestingBasis in Temp_ADPTTest table and evaluate
'TestingSalary, TestingContrib, and TestingMatch hold the values that will be used for testing. TestingBasis
' determines which values are stored and used in the testing calculations.
Select Case DLookup("TestingBasis", "Temp_ADPTTest", "")
    'Actual Contributions & Earnings
    Case 1
        'Update the testing fields in Temp_ADPTTest with YTDCompensation, Contributions, and Matchings
        db.Execute "UPDATE Temp_ADPTTest SET TestingValue = Value, TestingSalary = YTDCompensation, TestingContrib = Contributions,
TestingMatch = Matchings;"
        'Estimated Contributions & Earnings
    Case 2
        'Update the test fields in Temp_ADPTTest with EstSalary, EstContrib, and EstMatch
        db.Execute "UPDATE Temp_ADPTTest SET TestingValue = EstValue, TestingSalary = EstSalary, TestingContrib = EstContrib, TestingMatch =
EstMatch;"
End Select

```

```

'Set ADP and ACP values to zero where TestingSalary less than or equal to zero
db.Execute "UPDATE Temp_ADPTTest SET ADP = 0, ACP = 0 Where TestingSalary <= 0;"
'For all records with TestingSalary greater than zero calculate ADP by dividing TestingContrib by the lesser of MaxADPBase or TestingSalary
' and ACP by dividing TestingMatch by the lesser of MaxADPBase or TestingSalary
db.Execute "UPDATE Temp_ADPTTest Set ADP = cfix4(TestingContrib / LesserOf(MaxADPBase, TestingSalary)), ACP = cfix4(TestingMatch /
LesserOf(MaxADPBase, TestingSalary)) Where TestingSalary < 0;"

```

```

'Count Employees in Temp_ADPTTest that are eligible and HCEmployee is true
Set rs = db.OpenRecordset("Select Count(EmployeeID) As NumHC From Temp_ADPTTest Where HCEmployee = True And Eligible = True;"),
dbOpenSnapshot)
'Save count of NumHC
vNumHC = nullto0(rs&NumHC)
'Count Employees in Temp_ADPTTest that are eligible and HCEmployee is false
Set rs = db.OpenRecordset("Select Count(EmployeeID) As NumNHC From Temp_ADPTTest Where HCEmployee = False And Eligible = True;"),
dbOpenSnapshot)
'Save count of NumNHC
vNumNHC = nullto0(rs&NumNHC)

```

```

'Add totals of ADP and ACP from Temp_ADPTTest for employees that are eligible and HCEmployee is true
Set rs = db.OpenRecordset("Select Sum(ADP) As TotHCADP, Sum(ACP) As TotHCACP From Temp_ADPTTest Where HCEmployee = True And
Eligible = True;", dbOpenSnapshot)

```

```

'Save total of TotHCADP
vTotHCADP = cfix4(nullto0(rs&TotHCADP))
'Save total of TotHCACP
vTotHCACP = cfix4(nullto0(rs&TotHCACP))
'Add totals of ADP and ACP from Temp_ADPTTest for employees that are eligible and HCEmployee is false
Set rs = db.OpenRecordset("Select Sum(ADP) As TotNHCADP, Sum(ACP) As TotNHCACP From Temp_ADPTTest Where HCEmployee = False
And Eligible = True;", dbOpenSnapshot)
'Save total of TotNHCADP
vTotNHCADP = cfix4(nullto0(rs&TotNHCADP))
'Save total of TotNHCACP
vTotNHCACP = cfix4(nullto0(rs&TotNHCACP))

'Add totals of testingContrib from Temp_ADPTTest for employees that are eligible and KeyEmployee is true
Set rs = db.OpenRecordset("Select Sum(TestingValue) As TotKey From Temp_ADPTTest Where KeyEmployee = True And Eligible = True;",
dbOpenSnapshot)
'Save TotKey
vTotKey = cfix2(nullto0(rs&TotKey))
'Add totals of testingContrib from Temp_ADPTTest for employees that are eligible and KeyEmployee is false
Set rs = db.OpenRecordset("Select Sum(TestingValue) As TotNonKey From Temp_ADPTTest Where KeyEmployee = False And Eligible = True;",
dbOpenSnapshot)
'Save TotNonKey
vTotNonKey = cfix2(nullto0(rs&TotNonKey))

'Close recordset
rs.Close

'Update Temp_ADPTTest table HCADP = TotHCADP/NumHC
'      NHCADP = TotNHCADP/NumNHC
'      HCACP = TotHCACP/NumHC
'      NHCACP = TotNHCACP/NumNHC
'      TotKey = TotKey
'      TotNonKey = TotNonKey
db.Execute "UPDATE Temp_ADPTTest SET HCADP = " & cfix4(ffixDivBy0(vTotHCADP, vNumHC)) & ", NHCADP = " &
cfix4(ffixDivBy0(vTotNHCADP, vNumNHC)) & ", HCACP = " & cfix4(ffixDivBy0(vTotHCACP, vNumHC)) & ", NHCACP = " &
cfix4(ffixDivBy0(vTotNHCACP, vNumNHC)) & ", TotKey = " & vTotKey & ", TotNonKey = " & vTotNonKey & ";"

'Set mouse to pointer
DoCmd.Hourglass False

End Sub

Sub pCalculateVesting(vEmployerID, vVestedDate)
' Comments : Calculate vesting percentage based on date and employee hire date
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Dim rs As Recordset
Dim rs1 As Recordset
Dim vDate1
Dim vDate2 As Date
Dim vYear1
Dim vYear2
Dim vYearDiff
Dim vYears As Integer
Dim vNumDays As Integer
Dim vNumSun As Integer
Dim vNumSat As Integer
Dim vTrustDate As Date

Dim vImmediateVesting As Boolean
Dim vVestingBasis As Integer
Dim vVestingDate As Integer
Dim vVesting0 As Single
Dim vVesting1 As Single

```

```
Dim vVesting2 As Single
Dim vVesting3 As Single
Dim vVesting4 As Single
Dim vVesting5 As Single
Dim vVesting6 As Single
```

```
Set db = DBEngine.Workspaces(0).Databases(0)
```

```
'Get record from EmployerData table for selected Employer
Set rs = db.OpenRecordset("Select Distinctrow * from EmployerData where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Save value of ImmediateVesting from EmployerData record
vImmediateVesting = rs.ImmediateVesting
'save value of TrustDate from EmployerData record
vTrustDate = rs.TrustDate
```

```
'Save value of Vesting 0 from EmployerData record
vVesting0 = rs[Vesting 0]
'Save value of Vesting 1 from EmployerData record
vVesting1 = rs[Vesting 1]
'Save value of Vesting 2 from EmployerData record
vVesting2 = rs[Vesting 2]
'Save value of Vesting 3 from EmployerData record
vVesting3 = rs[Vesting 3]
'Save value of Vesting 4 from EmployerData record
vVesting4 = rs[Vesting 4]
'Save value of Vesting 5 from EmployerData record
vVesting5 = rs[Vesting 5]
'Save value of Vesting 6 from EmployerData record
vVesting6 = rs[Vesting 6]
'Save value of VestingBasis from EmployerData record
vVestingBasis = rs.VestingBasis
'Save value of VestingDate from EmployerData record
vVestingDate = rs.VestingDate
```

```
'Close recordset
rs.Close
```

```
'Get employee records from EmployeeData that match selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get all contribution records from ContributionData that match selected EmployerID
Set rs1 = db.OpenRecordset("Select Distinctrow * From ContributionData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Read all of Employee records in recordset just created until EOF
Do Until rs.EOF
```

```
    'If Immediate vesting is true then update
    If vImmediateVesting = True Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_Vested to 1
        rs.Temp_Vested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If
```

```
    'If employee is 65 or older then update
    If fGetAge(rs.EmployeeID) <= 65 Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_Vested to 1
        rs.Temp_Vested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If
```

```

'If employee has no hire date and vesting is based on hire date then update
If vVestingDate = 1 And IsNull(rs&[Hire Date]) Then
    'Prepare record for update
    rs.Edit
    'Set field Temp_Vested to 0
    rs&Temp_Vested = 0
    'Write record
    rs.Update
    'Skip all next logic to read next record
    GoTo PercentVested_NextRecord
End If

```

```

'If company has no plan date and vesting is based on plan date then update
If vVestingDate = 2 And IsNull(vTrustDate) Then
    'Prepare record for update
    rs.Edit
    'Set field Temp_Vested to 0
    rs&Temp_Vested = 0
    'Write record
    rs.Update
    'Skip all next logic to read next record
    GoTo PercentVested_NextRecord
End If

```

```

'If employee is not eligible and not terminated then update
If (rs&Eligible = False) And IsNull(rs&[Term Date]) Then
    'Prepare record for update
    rs.Edit
    'Set field Temp_Vested to 0
    rs&Temp_Vested = 0
    'Write record
    rs.Update
    'Skip all next logic to read next record
    GoTo PercentVested_NextRecord
End If

```

```

'If employee is not eligible and has been terminated
If (rs&Eligible = False) And Not IsNull(rs&[Term Date]) Then
    'Get first record in ContributionData for selected employee
    rs1.FindFirst "EmployeeID = " & rs&EmployeeID
    'If no record found then update
    If rs1.NoMatch Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_Vested to 0
        rs&Temp_Vested = 0
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If
End If

```

```

'Save employee hire date or trust (plan) date
If vVestingDate = 1 Then vDate1 = rs&[Hire Date] Else vDate1 = vTrustDate
'Save vested date
vDate2 = vVestedDate
'If employee has termination date overlay the vested date with the termination date
If Not IsNull(rs&[Term Date]) Then vDate2 = rs&[Term Date]
'Initialize to 0
vYears = 0
'Save the year from hire date
vYear1 = Year(vDate1)
'Save the year from vested date
vYear2 = Year(vDate2)
'Save the difference of the vested date and the hire date
vYearDiff = vYear2 - vYear1
'Evaluate the vesting basis
Select Case vVestingBasis

```

Case 1

'At least 1 year difference in years

If vYearDiff < 0 Then

'Reduce years by 1

vYears = vYearDiff - 1

'Save the number of days between the hire date and the end of that year

vNumDays = DateDiff("d", vDate1, "12/31/" & Str\$(vYear1))

'Save the number of weeks between the hire date and the end of that year with Sunday the first day of week

vNumSun = DateDiff("ww", vDate1, "12/31/" & Str\$(vYear1), 1)

'Save the number of weeks between the hire date and the end of that year with Saturday the first day of week

vNumSat = DateDiff("ww", vDate1, "12/31/" & Str\$(vYear1), 7)

'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday

vNumDays = vNumDays - vNumSun - vNumSat

'If the number of days equal or greater than 130 add 1 to years

If (vNumDays <= 130) Then vYears = vYears + 1

'Save the number of days from the first of the vested year to the vested date

vNumDays = DateDiff("d", "1/1/" & Str\$(vYear2), vDate2)

'Save the number of weeks between the first of the vested date and the vested date with Sunday the first day of week

vNumSun = DateDiff("ww", "1/1/" & Str\$(vYear2), vDate2, 1)

'Save the number of weeks between the first of the vested date and the vested date with Saturday the first day of week

vNumSat = DateDiff("ww", "1/1/" & Str\$(vYear2), vDate2, 7)

'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday

vNumDays = vNumDays - vNumSun - vNumSat

'If the number of days equal or greater than 130 add 1 to years

If (vNumDays <= 130) Then vYears = vYears + 1

'Same year hire date and vested date

Else

'Save the number of days between the hire date and the vested date

vNumDays = DateDiff("d", vDate1, vDate2)

'Save the number of weeks between the hire date and the vested date with Sunday the first day of week

vNumSun = DateDiff("ww", vDate1, vDate2, 1)

'Save the number of weeks between the hire date and the vested date with Saturday the first day of week

vNumSat = DateDiff("ww", vDate1, vDate2, 7)

'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday

vNumDays = vNumDays - vNumSun - vNumSat

'If the number of days equal or greater than 130 add 1 to years

If (vNumDays <= 130) Then vYears = vYears + 1

End If

Case 2

'Save the number of days between the hire date and the vested date

vNumDays = DateDiff("d", vDate1, vDate2)

'Save the number of weeks between the hire date and the vested date with Sunday the first day of week

vNumSun = DateDiff("ww", vDate1, vDate2, 1)

'Save the number of weeks between the hire date and the vested date with Saturday the first day of week

vNumSat = DateDiff("ww", vDate1, vDate2, 7)

'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday

vNumDays = vNumDays - vNumSun - vNumSat

'Save the result of integer division of number of days and 260

vYears = Int((vNumDays) \ 260)

'If the remainder of dividing number of days by 260 is equal or greater than 130 add 1 to years

If ((vNumDays Mod 260) <= 130) Then vYears = vYears + 1

End Select

'Vesting only goes out to max of 6 years

If vYears < 6 Then vYears = 6

'Prepare record for update

rs.Edit

'Evaluate the number of years vested and move percentage into Temp_Vested table

Select Case vYears

Case 0

'Move vested for 0

rs&Temp_Vested = vVesting0

Case 1

'Move vested for 1

rs&Temp_Vested = vVesting1

Case 2

'Move vested for 2

```

        rs&Temp_Vested = vVesting2
    Case 3
        'Move vested for 3
        rs&Temp_Vested = vVesting3
    Case 4
        'Move vested for 4
        rs&Temp_Vested = vVesting4
    Case 5
        'Move vested for 5
        rs&Temp_Vested = vVesting5
    Case 6
        'Move vested for 6
        rs&Temp_Vested = vVesting6
End Select
'Write record
rs.Update

PercentVested_NextRecord:
    'Move to next record
    rs.MoveNext
'Go back to top of Exit for next employee
Exit

'Close recordset of employees
rs.Close
'Close record set of ContributionData
rs1.Close

'pCalculateVesting_QNE vEmployerID, vVestedDate
pCalculateVesting_DECPS vEmployerID, vVestedDate

End Sub

Sub pCalculateVesting_QNE(vEmployerID, vVestedDate)
' Comments : Calculate vesting percentage based on date and employee hire date for QNE
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented :
' Modified :
'
' _____

Dim db As Database
Dim rs As Recordset
Dim rs1 As Recordset
Dim vDate1
Dim vDate2 As Date
Dim vYear1
Dim vYear2
Dim vYearDiff
Dim vYears As Integer
Dim vNumDays As Integer
Dim vNumSun As Integer
Dim vNumSat As Integer
Dim vTrustDate As Date

Dim vImmediateVesting As Boolean
Dim vVestingBasis As Integer
Dim vVestingDate As Integer
Dim vVesting0 As Single
Dim vVesting1 As Single
Dim vVesting2 As Single
Dim vVesting3 As Single
Dim vVesting4 As Single
Dim vVesting5 As Single
Dim vVesting6 As Single

Set db = DBEngine.Workspaces(0).Databases(0)

```

```

'Get record from EmployerData table for selected Employer
Set rs = db.OpenRecordset("Select Distinctrow * from EmployerData where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Save value of ImmediateVesting from EmployerData record
vImmediateVesting = rs&QNE_ImmediateVesting
'save value of TrustDate from EmployerData record
vTrustDate = rs&TrustDate

'Save value of Vesting 0 from EmployerData record
vVesting0 = rs&[QNE_Vesting 0]
'Save value of Vesting 1 from EmployerData record
vVesting1 = rs&[QNE_Vesting 1]
'Save value of Vesting 2 from EmployerData record
vVesting2 = rs&[QNE_Vesting 2]
'Save value of Vesting 3 from EmployerData record
vVesting3 = rs&[QNE_Vesting 3]
'Save value of Vesting 4 from EmployerData record
vVesting4 = rs&[QNE_Vesting 4]
'Save value of Vesting 5 from EmployerData record
vVesting5 = rs&[QNE_Vesting 5]
'Save value of Vesting 6 from EmployerData record
vVesting6 = rs&[QNE_Vesting 6]
'Save value of VestingBasis from EmployerData record
vVestingBasis = rs&QNE_VestingBasis
'Save value of VestingDate from EmployerData record
vVestingDate = rs&QNE_VestingDate

'Close recordset
rs.Close

'Get employee records from EmployeeData that match selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get all contribution records from ContributionData that match selected EmployerID
'Set rs1 = db.OpenRecordset("Select Distinctrow * From ContributionData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Read all of Employee records in recordset just created until EOF
Do Until rs.EOF

    'If Immediate vesting is true then update
    If vImmediateVesting = True Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_QNEVested to 1
        rs&Temp_QNEVested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If

    'If employee is 65 or older then update
    If fGetAge(rs&EmployeeID) <= 65 Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_QNEVested to 1
        rs&Temp_QNEVested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If

    'If employee has no hire date and vesting is based on hire date then update
    If vVestingDate = 1 And IsNull(rs&[Hire Date]) Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_QNEVested to 0
        rs&Temp_QNEVested = 0
        'Write record

```

```

rs.Update
'Skip all next logic to read next record
GoTo PercentVested_NextRecord
End If

'If company has no plan date and vesting is based on plan date then update
If vVestingDate = 2 And IsNull(vTrustDate) Then
  'Prepare record for update
  rs.Edit
  'Set field Temp_QNEVested to 0
  rs&Temp_QNEVested = 0
  'Write record
  rs.Update
  'Skip all next logic to read next record
  GoTo PercentVested_NextRecord
End If

```

'FOR QNE IT DOESN'T MATTER IF EMPLOYEE IS ELIGIBLE OR NOT, TERMINATED OR NOT, ETC.
'THEREFORE THE NEXT TWO EXITS HAVE BEEN DISABLED.

```

'If employee is not eligible and not terminated then update
'If (rs&Eligible = False) And IsNull(rs&[Term Date]) Then
  'Prepare record for update
  rs.Edit
  'Set field Temp_QNEVested to 0
  rs&Temp_QNEVested = 0
  'Write record
  rs.Update
  'Skip all next logic to read next record
  GoTo PercentVested_NextRecord
'End If

```

```

'If employee is not eligible and has been terminated
'If (rs&Eligible = False) And Not IsNull(rs&[Term Date]) Then
  'Get first record in ContributionData for selected employee
  rs1.FindFirst "EmployeeID = " & rs&EmployeeID
  'If no record found then update
  If rs1.NoMatch Then
    'Prepare record for update
    rs.Edit
    'Set field Temp_QNEVested to 0
    rs&Temp_QNEVested = 0
    'Write record
    rs.Update
    'Skip all next logic to read next record
    GoTo PercentVested_NextRecord
  End If
'End If

```

```

'Save employee hire date or trust (plan) date
If vVestingDate = 1 Then vDate1 = rs&[Hire Date] Else vDate1 = vTrustDate
'Save vested date
vDate2 = vVestedDate
'If employee has termination date overlay the vested date with the termination date
If Not IsNull(rs&[Term Date]) Then vDate2 = rs&[Term Date]
'Initialize to 0
vYears = 0
'Save the year from hire date
vYear1 = Year(vDate1)
'Save the year from vested date
vYear2 = Year(vDate2)
'Save the difference of the vested date and the hire date
vYearDiff = vYear2 - vYear1
'Evaluate the vesting basis
Select Case vVestingBasis
Case 1
  'At least 1 year difference in years
  If vYearDiff < 0 Then
    'Reduce years by 1
    vYears = vYearDiff - 1

```

```

'Save the number of days between the hire date and the end of that year
vNumDays = DateDiff("d", vDate1, "12/31/" & Str$(vYear1))
'Save the number of weeks between the hire date and the end of that year with Sunday the first day of week
vNumSun = DateDiff("ww", vDate1, "12/31/" & Str$(vYear1), 1)
'Save the number of weeks between the hire date and the end of that year with Saturday the first day of week
vNumSat = DateDiff("ww", vDate1, "12/31/" & Str$(vYear1), 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday
vNumDays = vNumDays - vNumSun - vNumSat
'If the number of days equal or greater than 130 add 1 to years
If (vNumDays <= 130) Then vYears = vYears + 1
'Save the number of days from the first of the vested year to the vested date
vNumDays = DateDiff("d", "1/1/" & Str$(vYear2), vDate2)
'Save the number of weeks between the first of the vested date and the vested date with Sunday the first day of week
vNumSun = DateDiff("ww", "1/1/" & Str$(vYear2), vDate2, 1)
'Save the number of weeks between the first of the vested date and the vested date with Saturday the first day of week
vNumSat = DateDiff("ww", "1/1/" & Str$(vYear2), vDate2, 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday
vNumDays = vNumDays - vNumSun - vNumSat
'If the number of days equal or greater than 130 add 1 to years
If (vNumDays <= 130) Then vYears = vYears + 1
'Same year hire date and vested date
Else
'Save the number of days between the hire date and the vested date
vNumDays = DateDiff("d", vDate1, vDate2)
'Save the number of weeks between the hire date and the vested date with Sunday the first day of week
vNumSun = DateDiff("ww", vDate1, vDate2, 1)
'Save the number of weeks between the hire date and the vested date with Saturday the first day of week
vNumSat = DateDiff("ww", vDate1, vDate2, 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday
vNumDays = vNumDays - vNumSun - vNumSat
'If the number of days equal or greater than 130 add 1 to years
If (vNumDays <= 130) Then vYears = vYears + 1
End If
Case 2
'Save the number of days between the hire date and the vested date
vNumDays = DateDiff("d", vDate1, vDate2)
'Save the number of weeks between the hire date and the vested date with Sunday the first day of week
vNumSun = DateDiff("ww", vDate1, vDate2, 1)
'Save the number of weeks between the hire date and the vested date with Saturday the first day of week
vNumSat = DateDiff("ww", vDate1, vDate2, 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday
vNumDays = vNumDays - vNumSun - vNumSat
'Save the result of integer division of number of days and 260
vYears = Int((vNumDays) \ 260)
'If the remainder of dividing number of days by 260 is equal or greater than 130 add 1 to years
If ((vNumDays Mod 260) <= 130) Then vYears = vYears + 1
End Select

'Vesting only goes out to max of 6 years
If vYears < 6 Then vYears = 6
'Prepare record for update
rs.Edit
'Evaluate the number of years vested and move percentage into Temp_QNEVested table
Select Case vYears
Case 0
'Move vested for 0
rs&Temp_QNEVested = vVesting0
Case 1
'Move vested for 1
rs&Temp_QNEVested = vVesting1
Case 2
'Move vested for 2
rs&Temp_QNEVested = vVesting2
Case 3
'Move vested for 3
rs&Temp_QNEVested = vVesting3
Case 4

```

```

        'Move vested for 4
        rs&Temp_QNEVested = vVesting4
    Case 5
        'Move vested for 5
        rs&Temp_QNEVested = vVesting5
    Case 6
        'Move vested for 6
        rs&Temp_QNEVested = vVesting6
    End Select
    'Write record
    rs.Update

PercentVested_NextRecord:
    'Move to next record
    rs.MoveNext
    'Go back to top of Exit for next employee
    Exit

    'Close recordset of employees
    rs.Close
    'Close record set of ContributionData
    'rs1.Close

End Sub

Sub pCalculateVesting_DECPS(vEmployerID, vVestedDate)
    ' Comments : Calculate vesting percentage based on date and employee hire date for DECPS
    ' Parameters : vEmployerID -
    ' Returns : -
    ' Created :
    ' Documented :
    ' Modified :
    '
    ' _____

    Dim db As Database
    Dim rs As Recordset
    Dim rs1 As Recordset
    Dim vDate1
    Dim vDate2 As Date
    Dim vYear1
    Dim vYear2
    Dim vYearDiff
    Dim vYears As Integer
    Dim vNumDays As Integer
    Dim vNumSun As Integer
    Dim vNumSat As Integer
    Dim vTrustDate As Date

    Dim vImmediateVesting As Boolean
    Dim vVestingBasis As Integer
    Dim vVestingDate As Integer
    Dim vVesting0 As Single
    Dim vVesting1 As Single
    Dim vVesting2 As Single
    Dim vVesting3 As Single
    Dim vVesting4 As Single
    Dim vVesting5 As Single
    Dim vVesting6 As Single

    Set db = DBEngine.Workspaces(0).Databases(0)

    'Get record from EmployerData table for selected Employer
    Set rs = db.OpenRecordset("Select Distinctrow * from EmployerData where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
    'Save value of ImmediateVesting from EmployerData record
    vImmediateVesting = rs&DECPS_ImmediateVesting
    'save value of TrustDate from EmployerData record
    vTrustDate = rs&TrustDate

```

```

'Save value of Vesting 0 from EmployerData record
vVesting0 = rs&[DECPS_Vesting 0]
'Save value of Vesting 1 from EmployerData record
vVesting1 = rs&[DECPS_Vesting 1]
'Save value of Vesting 2 from EmployerData record
vVesting2 = rs&[DECPS_Vesting 2]
'Save value of Vesting 3 from EmployerData record
vVesting3 = rs&[DECPS_Vesting 3]
'Save value of Vesting 4 from EmployerData record
vVesting4 = rs&[DECPS_Vesting 4]
'Save value of Vesting 5 from EmployerData record
vVesting5 = rs&[DECPS_Vesting 5]
'Save value of Vesting 6 from EmployerData record
vVesting6 = rs&[DECPS_Vesting 6]
'Save value of VestingBasis from EmployerData record
vVestingBasis = rs&DECPS_VestingBasis
'Save value of VestingDate from EmployerData record
vVestingDate = rs&DECPS_VestingDate

'Close recordset
rs.Close

'Get employee records from EmployeeData that match selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get all contribution records from ContributionData that match selected EmployerID
'Set rs1 = db.OpenRecordset("Select Distinctrow * From ContributionData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Read all of Employee records in recordset just created until EOF
Do Until rs.EOF

    'If Immediate vesting is true then update
    If vImmediateVesting = True Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_DECPSVested to 1
        rs&Temp_DECPSVested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If

    'If employee is 65 or older then update
    If fGetAge(rs&EmployeeID) <= 65 Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_DECPSVested to 1
        rs&Temp_DECPSVested = 1
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If

    'If employee has no hire date and vesting is based on hire date then update
    If vVestingDate = 1 And IsNull(rs&[Hire Date]) Then
        'Prepare record for update
        rs.Edit
        'Set field Temp_DECPSVested to 0
        rs&Temp_DECPSVested = 0
        'Write record
        rs.Update
        'Skip all next logic to read next record
        GoTo PercentVested_NextRecord
    End If

    'If company has no plan date and vesting is based on plan date then update
    If vVestingDate = 2 And IsNull(vTrustDate) Then
        'Prepare record for update

```

```

rs.Edit
'Set field Temp_DECPSVested to 0
rs&Temp_DECPSVested = 0
'Write record
rs.Update
'Skip all next logic to read next record
GoTo PercentVested_NextRecord
End If

```

```

'FOR QNE IT DOESN'T MATTER IF EMPLOYEE IS ELIGIBLE OR NOT, TERMINATED OR NOT, ETC.
'THEREFORE THE NEXT TWO EXITS HAVE BEEN DISABLED.

```

```

'If employee is not eligible and not terminated then update
'If (rs&Eligible = False) And IsNull(rs&[Term Date]) Then
'   'Prepare record for update
'   rs.Edit
'   'Set field Temp_DECPSVested to 0
'   rs&Temp_DECPSVested = 0
'   'Write record
'   rs.Update
'   'Skip all next logic to read next record
'   GoTo PercentVested_NextRecord
'End If

```

```

'If employee is not eligible and has been terminated
'If (rs&Eligible = False) And Not IsNull(rs&[Term Date]) Then
'   'Get first record in ContributionData for selected employee
'   rs1.FindFirst "EmployeeID = " & rs&EmployeeID
'   'If no record found then update
'   If rs1.NoMatch Then
'       'Prepare record for update
'       rs.Edit
'       'Set field Temp_DECPSVested to 0
'       rs&Temp_DECPSVested = 0
'       'Write record
'       rs.Update
'       'Skip all next logic to read next record
'       GoTo PercentVested_NextRecord
'   End If
'End If

```

```

'Save employee hire date or trust (plan) date
If vVestingDate = 1 Then vDate1 = rs&[Hire Date] Else vDate1 = vTrustDate
'Save vested date
vDate2 = vVestedDate
'If employee has termination date overlay the vested date with the termination date
If Not IsNull(rs&[Term Date]) Then vDate2 = rs&[Term Date]
'Initialize to 0
vYears = 0
'Save the year from hire date
vYear1 = Year(vDate1)
'Save the year from vested date
vYear2 = Year(vDate2)
'Save the difference of the vested date and the hire date
vYearDiff = vYear2 - vYear1
'Evaluate the vesting basis
Select Case vVestingBasis
Case 1
'At least 1 year difference in years
If vYearDiff < 0 Then
'Reduce years by 1
vYears = vYearDiff - 1
'Save the number of days between the hire date and the end of that year
vNumDays = DateDiff("d", vDate1, "12/31/" & Str$(vYear1))
'Save the number of weeks between the hire date and the end of that year with Sunday the first day of week
vNumSun = DateDiff("ww", vDate1, "12/31/" & Str$(vYear1), 1)
'Save the number of weeks between the hire date and the end of that year with Saturday the first day of week
vNumSat = DateDiff("ww", vDate1, "12/31/" & Str$(vYear1), 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday

```

```

vNumDays = vNumDays - vNumSun - vNumSat
'If the number of days equal or greater than 130 add 1 to years
If (vNumDays <= 130) Then vYears = vYears + 1
'Save the number of days from the first of the vested year to the vested date
vNumDays = DateDiff("d", "1/1/" & Str$(vYear2), vDate2)
'Save the number of weeks between the first of the vested date and the vested date with Sunday the first day of week
vNumSun = DateDiff("ww", "1/1/" & Str$(vYear2), vDate2, 1)
'Save the number of weeks between the first of the vested date and the vested date with Saturday the first day of week
vNumSat = DateDiff("ww", "1/1/" & Str$(vYear2), vDate2, 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with Saturday
vNumDays = vNumDays - vNumSun - vNumSat
'If the number of days equal or greater than 130 add 1 to years
If (vNumDays <= 130) Then vYears = vYears + 1
'Same year hire date and vested date
Else
'Save the number of days between the hire date and the vested date
vNumDays = DateDiff("d", vDate1, vDate2)
'Save the number of weeks between the hire date and the vested date with Sunday the first day of week
vNumSun = DateDiff("ww", vDate1, vDate2, 1)
'Save the number of weeks between the hire date and the vested date with Saturday the first day of week
vNumSat = DateDiff("ww", vDate1, vDate2, 7)
'Save the number of days less the number of weeks starting with Sunday and the number of weeks starting with
rs.Close
'Close record set of ContributionData
'rs1.Close

End Sub

Public Sub pCheckEligibilityAll(vEmployerID)
' Comments : Determine employee eligibility based on employer criteria
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Dim rs As Recordset
Dim vEligibilityAge As Integer
Dim vEligibilityMonths As Integer
Dim vEligibilityPartTimers As Boolean
Dim vEligibilityUnionMembers As Boolean
Dim vEligibilityNonResidentAliens As Boolean
Dim vBirthDate As Date
Dim vHireDate As Date
Dim vTerminated As Boolean
Dim vPartTimeEmployee As Boolean
Dim vUnionMember As Boolean
Dim vNonResidentAlien As Boolean

Set db = DBEngine.Workspaces(0).Databases(0)
'Get employer record for selected employerID
Set rs = db.OpenRecordset("Select Distinctrow * From EmployerData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'If there is no eligibility age or eligible months then display message to enter this information
If IsNull(rs&EligibilityAge) Or IsNull(rs&EligibilityMonths) Then
'Display message to user to enter eligiility information
MsgBox "Please enter eligibility information.", vbInformation, "Update Eligibility"
'Close recordset
rs.Close
'Exit procedure
Exit Sub
End If
'Save eligibility age
vEligibilityAge = rs&EligibilityAge
'Save eligibility months
vEligibilityMonths = rs&EligibilityMonths

```

```

'Save eligibility part time
vEligibilityPartTimers = rs&EligibilityPartTimers
'Save eligibility union members
vEligibilityUnionMembers = rs&EligibilityUnionMembers
'Save eligibility non resident aliens
vEligibilityNonResidentAliens = rs&EligibilityNonResidentAliens
'Close recordset
rs.Close

'Get all employees from EmployeeData for selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * From EmployeeData Where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Read all employees records
Do Until rs.EOF

    'If the birth date or hire date is blank mark as ineligible
    If IsNull(rs&BirthDate) Or IsNull(rs&[Hire Date]) Then
        'Prepare record for update
        rs.Edit
        'Mark Eligible false
        rs&Eligible = False
        'Write record
        rs.Update
    Else
        'Save birth date
        vBirthDate = rs&BirthDate
        'Save hire date
        vHireDate = rs&[Hire Date]
        'Save terminated value
        If rs&[Terminated] = "Terminated" Then vTerminated = True Else vTerminated = False
        'Save part time employee
        vPartTimeEmployee = rs&PartTimeEmployee
        'Save union member
        vUnionMember = rs&UnionMember
        'Save non resident alien
        vNonResidentAlien = rs&NonResidentAlien
        'Prepare record for update
        rs.Edit
        'If employee is terminated
        If vTerminated = True Then
            'Eligible is false
            rs&Eligible = False
        'If union members are not eligible and employee is union member
        ElseIf vEligibilityUnionMembers = False And vUnionMember = True Then
            'Eligible is false
            rs&Eligibility = False
        'If non resident aliens are not eligible and employee is non resident alien
        ElseIf vEligibilityNonResidentAliens = False And vNonResidentAlien = True Then
            'Eligible is false
            rs&Eligibility = False
        'If prt timers are not eligible and member is part time
        ElseIf vEligibilityPartTimers = False And vPartTimeEmployee = True Then
            'Eligible is false
            rs&Eligible = False
        'If employee is less than eligibility age
        ElseIf vEligibilityAge < fGetAge(rs&EmployeeID) Then
            'Eligible is false
            rs&Eligible = False
        'If eligibility months required are more than employee has worked
        ElseIf vEligibilityMonths < fGetMonths(vHireDate, Date) Then
            'Eligible is false
            rs&Eligible = False
        Else
            'Eligible is true
            rs&Eligible = True
        End If
        'Write record
        rs.Update
    End If
'Move to next employee

```

```
rs.MoveNext
'Go back to top for next employee
Exit
'Close recordset
rs.Close
```

End Sub

```
Public Sub pClearTempADPData()
' Comments : Clear records from temp_ADPTTest table
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Set db = DBEngine.Workspaces(0).Databases(0)
'Delete all records from Temp_ADPTTest table
db.Execute "Delete Distinctrow * From Temp_ADPTTest;"
End Sub
```

```
Public Sub pClearTempBatchData()
' Comments : Clear all records from temporary batch tables
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Set db = DBEngine.Workspaces(0).Databases(0)

'Delete all records from Temp_BatchData
db.Execute "Delete Distinctrow * From Temp_BatchData;"
'Delete all records from temp_BatchMF
db.Execute "Delete Distinctrow * From Temp_BatchMF;"
End Sub
```

```
Public Sub pClearTempVestingData()
' Comments : Delete all records from Temp_VestingReport work table
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Set db = DBEngine.Workspaces(0).Databases(0)
'Delete all records from TempVestingReport table
db.Execute "Delete Distinctrow * From Temp_VestingReport;"
End Sub
```

```
Public Sub pCreateMFApp(vType, vEmployerID, vPeriod, vBatchID, vActivityID)
' Comments : Build temporary table Temp_MFApp to process desired groups of records
' Parameters : vType
' vPeriod
' vActivityID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim vDB As Database
Dim vRS As Recordset
```

```
Dim vQueryStr As String
Dim vEmployerTaxID As String
Set vDB = DBEngine.Workspaces(0).Databases(0)
```

'Evaluate selected Type to determine what records are desired in new table. AllocatedFunds table supplies the records

Select Case vType

'Want all records for selected EmployerID and period, ContributionID not equal zero and have no AccountNum

Case 1 'Monthly Processing

vQueryStr = "INSERT INTO Temp_MFApp (EmployerID, EmployeeID, PortfolioID, Amount) "

vQueryStr = vQueryStr & "SELECT EmployerID, EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "

vQueryStr = vQueryStr & "FROM AllocatedFunds "

vQueryStr = vQueryStr & "WHERE ((IsNull(AccountNum)) And (EmployerID = " & vEmployerID & ") And (Period = " & vPeriod & ") And (ContributionID << 0)) "

vQueryStr = vQueryStr & "GROUP BY EmployerID, EmployeeID, PortfolioID;"

'Want all records for selected BatchID and no AccountNum

Case 2 'Batch Processing

vQueryStr = "INSERT INTO Temp_MFApp (EmployerID, EmployeeID, PortfolioID, Amount) "

vQueryStr = vQueryStr & "SELECT EmployerID, EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "

vQueryStr = vQueryStr & "FROM AllocatedFunds "

vQueryStr = vQueryStr & "WHERE (IsNull(AccountNum) And (BatchID = " & vBatchID & ")) "

vQueryStr = vQueryStr & "GROUP BY EmployerID, EmployeeID, PortfolioID;"

'Want all records for selected ActivityID

Case 3 'Individual Item

vQueryStr = "INSERT INTO Temp_MFApp (EmployerID, EmployeeID, PortfolioID, Amount) "

vQueryStr = vQueryStr & "SELECT EmployerID, EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "

vQueryStr = vQueryStr & "FROM AllocatedFunds "

vQueryStr = vQueryStr & "WHERE ((ActivityID = " & vActivityID & ")) "

vQueryStr = vQueryStr & "GROUP BY EmployerID, EmployeeID, PortfolioID;"

End Select

'Delete all records from Temp_MFApp table

vDB.Execute "Delete * From Temp_MFApp;"

'Build new Temp_MFApp table with SQL string just created above

vDB.Execute vQueryStr

'Update Temp_MFApp table with ReportTaxID equal to employee social security number for all records with MFAppTaxID = 2

vQueryStr = "UPDATE ((Temp_MFApp INNER JOIN Portfolios ON Temp_MFApp.PortfolioID = Portfolios.PortfolioID) INNER JOIN FundFamilies ON Portfolios.MutualFundID = FundFamilies.MutualFundID) INNER JOIN EmployeeData ON Temp_MFApp.EmployeeID = EmployeeData.EmployeeID SET Temp_MFApp.ReportTaxID = [EmployeeData].[SSNum] "

vQueryStr = vQueryStr & "WHERE (((FundFamilies.MFAppTaxID)=2));"

vDB.Execute vQueryStr

'Update Temp_MFApp table with ReportTaxID equal to employer tax ID number for all records with MFAppTaxID = 1

Set vRS = vDB.OpenRecordset("Select [Tax ID] FROM EmployerData;", dbOpenSnapshot)

vEmployerTaxID = fDecrypt(vRS("Tax ID"))

vRS.Close

Set vRS = Nothing

vQueryStr = "UPDATE ((Temp_MFApp INNER JOIN Portfolios ON Temp_MFApp.PortfolioID = Portfolios.PortfolioID) INNER JOIN FundFamilies ON Portfolios.MutualFundID = FundFamilies.MutualFundID) INNER JOIN EmployerData ON Temp_MFApp.EmployeeID = EmployerData.EmployeeID SET Temp_MFApp.ReportTaxID = " & vEmployerTaxID & " "

vQueryStr = vQueryStr & "WHERE (((FundFamilies.MFAppTaxID)=1));"

vDB.Execute vQueryStr

End Sub

Public Sub pCreateTempADPData(vEmployerID, vTestYear, vHCBasis, vTop20Switch)

' Comments : Appends the next group of records to be processed to the Temp_ADPTTest table. Employee records

' are created and contributions and matching amounts for eligible employees are processed for

' the selected employer and period and uses the HCBasis and Top20Switch values

' Parameters : vEmployerID

' vTestYear -

' vHCBasis -

' vTop20Switch -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

' _____

```

Dim db As Database
Dim rs
Dim rs1 As Recordset
Dim vCompanyName
Dim vQueryStr
Dim vSourceStr As String
Dim vMaxADPBase As Currency
Dim vMatching As Boolean
Dim vImmediateVesting As Boolean
Set db = DBEngine.Workspaces(0).Databases(0)

'Set mouse to hourglass
DoCmd.Hourglass True

'Select employer record from EmployerData
Set rs = db.OpenRecordset("Select Distinctrow * From EmployerData Where EmployerID = " & vEmployerID & ";", dbOpenSnapshot)
'Save company name
vCompanyName = rs&Company
'Save matching
vMatching = rs&Matching
'Save Immediate Vesting
vImmediateVesting = rs&ImmediateVesting
'If Matching is true and ImmediateVesting is true
If (rs&Matching = True) And (rs&ImmediateVesting = True) Then
    'Build a string to modify a where clause for Source = 1,2,10, and 14
    vSourceStr = "(AllocatedFunds.Source = 1) or (AllocatedFunds.Source = 2) or (AllocatedFunds.Source = 10) or (AllocatedFunds.Source = 14)"
Else
    'Build a string to modify a where clause for Source = 1,10, and 14
    vSourceStr = "(AllocatedFunds.Source = 1) or (AllocatedFunds.Source = 10) or (AllocatedFunds.Source = 14)"
End If
'Build recordset from T_ComplianceTesting for selected test year
Set rs = db.OpenRecordset("Select Distinctrow * From T_ComplianceTesting Where TestingYear = " & vTestYear & ";", dbOpenSnapshot)
'Save MaxADPBase from first record
vMaxADPBase = rs&MaxADPBase
'close recordset
rs.Close

'Append employees to Temp_ADPTTest table from EmployeeData that are still working or were not terminated before test year for selected
EmployerID
vQueryStr = "INSERT INTO Temp_ADPTTest ( EmployeeID ) "
vQueryStr = vQueryStr & "SELECT EmployeeData.EmployeeID FROM EmployeeData "
vQueryStr = vQueryStr & "WHERE (((EmployeeData.EmployeeID)=" & vEmployerID & ") AND (Year(EmployeeData.[Hire Date]) <= " &
vTestYear & ") AND ((EmployeeData.Terminated Is Null) or (Year(EmployeeData.[Term Date]) <= " & vTestYear & "));"
db.Execute vQueryStr

'Append employees and their contributions to Temp_ADPTTest table from AllocatedFunds for selected EmployerID, period = test year, and the
Source types saved
vQueryStr = "INSERT INTO Temp_ADPTTest ( EmployeeID, Contributions ) "
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((AllocatedFunds.EmployeeID = " & vEmployerID & ") And (Int(AllocatedFunds.Period) = " & vTestYear & ")
And (" & vSourceStr & ") "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr

'Add EstValue for top-heavy test. This is for all sources, inception-to-date.
vQueryStr = "INSERT INTO Temp_ADPTTest ( EmployeeID, EstValue ) "
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((AllocatedFunds.EmployeeID = " & vEmployerID & ") "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr

'If Matching is true
If vMatching = True Then
    'Append employees and their matching amounts to Temp_ADPTTest table from AllocatedFunds for selected EmployerID, period = test year, and
the Source type 2

```

```

vQueryStr = "INSERT INTO Temp_ADPTTest ( EmployeeID, Matchings )"
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((AllocatedFunds.EmployeeID = " & vEmployerID & ") And (Int(AllocatedFunds.Period) = " & vTestYear &
") And (AllocatedFunds.Source = 2)) "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr
End If

'Append employees and their total contribution amounts, total matching amounts and a SummaryRecord to Temp_ADPTTest table from
Temp_ADPTTest
vQueryStr = "INSERT INTO Temp_ADPTTest ( EmployeeID, Contributions, Matchings, EstValue, SummaryRecord )"
vQueryStr = vQueryStr & "SELECT Temp_ADPTTest.EmployeeID, Sum(Temp_ADPTTest.Contributions) AS SumOfContributions,
Sum(Temp_ADPTTest.Matchings) AS SumOfMatchings, Sum(Temp_ADPTTest.EstValue) AS SumOfEstValue, True AS Expr1 "
vQueryStr = vQueryStr & "FROM Temp_ADPTTest "
vQueryStr = vQueryStr & "GROUP BY Temp_ADPTTest.EmployeeID;"
db.Execute vQueryStr

'Delete all of the new records appended to Temp_ADPTTest except for the summary record
db.Execute "Delete Distinctrow * From Temp_ADPTTest Where SummaryRecord = False;"
'Update all records in Temp_ADPTTest for selected matching and immediate vesting values
db.Execute "UPDATE Temp_ADPTTest SET Temp_ADPTTest.Matching = " & vMatching & ", Temp_ADPTTest.ImmediateVesting = " &
vImmediateVesting & ";"
'Update all records in Temp_ADPTTest for test year, EmployerID, company name, MaxADPBase, HCBasis, Top20Switch, and testing basis = 1
db.Execute "UPDATE Temp_ADPTTest SET Temp_ADPTTest.Year = " & vTestYear & ", Temp_ADPTTest.EmployeeID = " & vEmployerID & ",
Temp_ADPTTest.Company = " & fDecrypt(vCompanyName) & ", Temp_ADPTTest.MaxADPBase = " & vMaxADPBase & ",
Temp_ADPTTest.HCBasis = " & vHCBasis & ", Temp_ADPTTest.Top20Switch = " & vTop20Switch & ", Temp_ADPTTest.TestingBasis = 1;"
'Update employee fields in Temp_ADPTTest from EmployeeData
db.Execute "UPDATE EmployeeData INNER JOIN Temp_ADPTTest ON EmployeeData.EmployeeID = Temp_ADPTTest.EmployeeID SET
Temp_ADPTTest.Value = [EmployeeData].[Temp_CurrentValue], Temp_ADPTTest.SSNum = [EmployeeData].[SSNum],
Temp_ADPTTest.EmployeeName = [EmployeeData].[Last Name] & ', ' & [EmployeeData].[First Name] & ' ' & [EmployeeData].[Middle Initial],
Temp_ADPTTest.EstSalary = [EmployeeData].[EstSalary], Temp_ADPTTest.PriorYearSalary = [EmployeeData].[PriorYearSalary],
Temp_ADPTTest.YTDCCompensation = [EmployeeData].[YTDCCompensation], Temp_ADPTTest.Eligible = [EmployeeData].[Eligible];"

'Create recordset from Temp_ADPTTest who are not eligible
Set rs = db.OpenRecordset("Select Distinctrow * From Temp_ADPTTest Where Eligible = False ORDER BY EmployeeID;", dbOpenDynaset)
'Create recordset from ContributionData for eligible employees for selected test year and EmployerID
Set rs1 = db.OpenRecordset("Select Distinctrow * From ContributionData Where Int(Period) = " & vTestYear & " AND EmployerID = " &
vEmployerID & " AND Eligible = True ORDER BY EmployeeID;", dbOpenSnapshot)
'Read through all records from Temp_ADPTTest
Do Until rs.EOF
'Look for match of employee in ContributionData recordset to see if eligible
rs1.FindNext "EmployeeID = " & rs&EmployeeID
'If find an employee match
If Not rs1.NoMatch Then
'Prepare record for update
rs.Edit
'Set Eligible to true
rs&Eligible = True
'Write record
rs.Update
End If
'Move to next record in Temp_ADPTTest recordset
rs.MoveNext
'Go back to process next employee record
Exit
'Close recordsets
rs1.Close
rs.Close

'Calculate contribution amounts
SetEstimatedContrib vEmployerID, vTestYear
'Calculate matching amounts
SetEstimatedMatch vEmployerID, vTestYear
'Set mouse to pointer
DoCmd.Hourglass False

```

End Sub

Public Sub pCreateTempBatchData(vEmployerID)

‘ Comments : Append Employee and EmployeeMF records to Temp_BatchData table for selected EmployerID

‘ Parameters : vEmployerID -

‘ Returns :

‘ Created :

‘ Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

‘ Modified :

‘

‘ _____

Dim db As Database

Dim vQueryStr As String

Set db = DBEngine.Workspaces(0).Databases(0)

‘Set mouse to hourglass

DoCmd.Hourglass True

‘Adds records to Temp_BatchData table using EmployeeID, EmployeeName, and SSNum from EmployeeData table for selected EmployerID

vQueryStr = "INSERT INTO Temp_BatchData (EmployeeID, EmployeeName, SSNum) "

vQueryStr = vQueryStr & "SELECT EmployeeData.EmployeeID, [Last Name] & ' ' & [First Name] & ' ' & [Middle Initial] AS Expr2, EmployeeData.SSNum "

vQueryStr = vQueryStr & "FROM EmployeeData "

vQueryStr = vQueryStr & "WHERE (((EmployeeData.EmployerID)=" & vEmployerID & ");"

db.Execute vQueryStr

‘Adds records to Temp_BatchData table using EmployeeID, PortfolioID, Allocation, and AccountNUM from EmployeeMF table for selected EmployerID

vQueryStr = "INSERT INTO Temp_BatchMF (EmployeeID, PortfolioID, Allocation, AccountNum) "

vQueryStr = vQueryStr & "SELECT EmployeeMF.EmployeeID, EmployeeMF.PortfolioID, EmployeeMF.Allocation, EmployeeMF.AccountNum "

vQueryStr = vQueryStr & "FROM EmployeeMF "

vQueryStr = vQueryStr & "WHERE (((EmployeeMF.EmployerID)=" & vEmployerID & ");"

db.Execute vQueryStr

‘Set mouse to pointer

DoCmd.Hourglass False

End Sub

Public Sub pCreateTempVestingData(vEmployerID, vDate)

‘ Comments : Appends records to Temp_VestingReport table for employees and their vested amounts

‘ Parameters : vEmployerID -

‘ Returns : -

‘ Created :

‘ Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

‘ Modified :

‘

‘ _____

Dim db As Database

Dim vQueryStr As String

Dim vEligibilityDate As String

Set db = DBEngine.Workspaces(0).Databases(0)

Select Case DLookup("VestingDate", "EmployerData", "")

Case 1

vEligibilityDate = "EmployeeData.[Hire Date]"

Case 2

vEligibilityDate = "EmployeeData.[TrustDate]"

End Select

‘Append all employee records to Temp_VestingReport table using EmployeeData for selected EmployerID

db.Execute "INSERT INTO Temp_VestingReport (EmployeeID) SELECT EmployeeData.EmployeeID FROM EmployeeData WHERE EmployeeData.EmployerID=" & vEmployerID & ";"

```

'Append records from AllocatedFunds table to Temp_VestingReport table using total amounts for source = 2 for selected EmployerID
vQueryStr = "INSERT INTO Temp_VestingReport ( EmployeeID, EmployerContribution ) "
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((AllocatedFunds.EmployeeID = " & vEmployerID & ") And (AllocatedFunds.Source = 2)) "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr

```

```

'Append records from AllocatedFunds table to Temp_VestingReport table using total amounts for source = 15 for selected EmployerID
vQueryStr = "INSERT INTO Temp_VestingReport ( EmployeeID, DECPS_Contribution ) "
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((AllocatedFunds.EmployeeID = " & vEmployerID & ") And (AllocatedFunds.Source = 15)) "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr

```

```

'Append records from Temp_VestingReport table to Temp_VestingReport table using total employer contribution amounts for all records, and in
for selected EmployerID
vQueryStr = "INSERT INTO Temp_VestingReport ( EmployeeID, EmployerContribution, DECPS_Contribution, SummaryRecord ) "
vQueryStr = vQueryStr & "SELECT Temp_VestingReport.EmployeeID, Sum(Temp_VestingReport.EmployerContribution) AS
SumOfEmployerContribution, Sum(Temp_VestingReport.DECPS_Contribution) AS SumOfDECPS_Contribution, True AS Expr1 "
vQueryStr = vQueryStr & "FROM Temp_VestingReport "
vQueryStr = vQueryStr & "GROUP BY Temp_VestingReport.EmployeeID;"
db.Execute vQueryStr

```

```

'Delete all records form Temp_VestingReport where Summary record is false
db.Execute "Delete Distinctrow * From Temp_VestingReport Where SummaryRecord = False;"

```

```

'Update all records with the selected EmployerID
db.Execute "UPDATE Temp_VestingReport SET Temp_VestingReport.EmployerID = " & vEmployerID & ";"

```

```

'Update all records with Company name from EmployerData table, and match on EmployeeID to fill in Name, social security number, hire date,
termination date, eligibility, and calculate vested amount equal to employer contributions times vesting
db.Execute "UPDATE (Temp_VestingReport INNER JOIN EmployeeData ON Temp_VestingReport.EmployeeID = EmployeeData.EmployeeID)
INNER JOIN EmployerData ON Temp_VestingReport.EmployerID = EmployerData.EmployerID SET Temp_VestingReport.Company =
[EmployerData].[Company], Temp_VestingReport.EmployeeName = [EmployeeData].[Last Name] & ', ' & [EmployeeData].[First Name] & ' ' &
[EmployeeData].[Middle Initial], Temp_VestingReport.SSNum = [EmployeeData].[SSNum], Temp_VestingReport.EligibilityDate = " &
vEligibilityDate & ", Temp_VestingReport.TermDate = [EmployeeData].[Term Date], Temp_VestingReport.Eligible = EmployeeData.Eligible,
Temp_VestingReport.VestingPercent = EmployeeData.Temp_Vested, Temp_VestingReport.DECPS_VestingPercent =
EmployeeData.Temp_DECPSVested, Temp_VestingReport.VestedAmount = (EmployeeData.Temp_Vested *
Temp_VestingReport.EmployerContribution), Temp_VestingReport.DECPS_VestedAmount = (EmployeeData.Temp_DECPSVested *
Temp_VestingReport.DECPS_Contribution);"

```

```

'Delete all records for employees who are not eligible and are not terminated
'db.Execute "Delete Distinctrow * From Temp_VestingReport Where Eligible = False And IsNull(TermDate);"

```

```

'Delete all records for employees who are not eligible and were terminated in the prior year
db.Execute "Delete Distinctrow * From Temp_VestingReport Where Eligible = False And Year(TermDate) < " & Year(vDate) & ";"

```

```

'Update all records with the AsOfDate equal to the selected date
db.Execute "UPDATE Temp_VestingReport SET Temp_VestingReport.AsOfDate = #" & vDate & "#;"

```

End Sub

Public Sub pCreateTempYTDActivityByFamily(vYear, vEmployerID)

' Comments : Create Temp_YTDActivity table with all YTD activity by source and mutual fund between a selected
' period plus one year and EmployerID

' Parameters : vYear -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

' _____

```
Dim db As Database
Dim vQueryStr As String
Set db = DBEngine.Workspaces(0).Databases(0)
```

```
'Delete all records from Temp_YTDActivity table
db.Execute "Delete * from Temp_YTDActivity;"
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all Contributions for selected year and next year, EmployerID, and
source = 1
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Contributions ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=1) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all matchings for selected year and next year, EmployerID, and source
= 2
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Matchings ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=2) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all loan payments for selected year and next year, EmployerID, and
source = 3
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, LoanPayments ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=3) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all rollovers for selected year and next year, EmployerID, and source =
5
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Rollovers ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=5) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all transfers for selected year and next year, EmployerID, and source =
6
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Transfers ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=6) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all loan distributions for selected year and next year, EmployerID, and
source = 7
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, LoanDistributions ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=7) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, MutualFundID and all Idistributions for selected year and next year, EmployerID, and
source = 9
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Distributions ) "
```

```

vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=9)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr

```

'Append records from AllocatedFunds for EmployeeID, MutualFundID and all other adjustments for selected year and next year, EmployerID, and source = 10 or 11

```

vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, OtherAdjustments ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=10 OR Source=11)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr

```

'Append records from AllocatedFunds for EmployeeID, MutualFundID and all non vested forfeitures for selected year and next year, EmployerID, and source = 12

```

vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, NonVestedForfeitures ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, MutualFundID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=12)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, MutualFundID;"
db.Execute vQueryStr

```

'Delete all records with EmployeeID is null or MutualFundID is null

```

vQueryStr = "DELETE Distinctrow * From Temp_YTDActivity Where IsNull(EmployeeID) or IsNull(MutualFundID);"
db.Execute vQueryStr

```

'Append summary records for all different types of amounts and set SummaryRecord to true

```

vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, MutualFundID, Contributions, Matchings, LoanPayments, Rollovers, Transfers, LoanDistributions, Distributions, OtherAdjustments, NonVestedForfeitures, SummaryRecord ) "
vQueryStr = vQueryStr & "SELECT Temp_YTDActivity.EmployeeID, Temp_YTDActivity.MutualFundID, Sum(Temp_YTDActivity.Contributions) AS SumOfContributions, Sum(Temp_YTDActivity.Matchings) AS SumOfMatchings, Sum(Temp_YTDActivity.LoanPayments) AS SumOfLoanPayments, Sum(Temp_YTDActivity.Rollovers) AS SumOfRollovers, Sum(Temp_YTDActivity.Transfers) AS SumOfTransfers, Sum(Temp_YTDActivity.LoanDistributions) AS SumOfLoanDistributions, Sum(Temp_YTDActivity.Distributions) AS SumOfDistributions, Sum(Temp_YTDActivity.OtherAdjustments) AS SumOfOtherAdjustments, Sum(Temp_YTDActivity.NonVestedForfeitures) AS SumOfNonVestedForfeitures, True AS Expr1 "
vQueryStr = vQueryStr & "FROM Temp_YTDActivity "
vQueryStr = vQueryStr & "GROUP BY Temp_YTDActivity.EmployeeID, Temp_YTDActivity.MutualFundID;"
db.Execute vQueryStr

```

'Delete all records that are not summary records

```

db.Execute "DELETE Distinctrow * FROM Temp_YTDActivity WHERE SummaryRecord = False;"

```

'Update all records with selected EmployerID and year

```

db.Execute "UPDATE Temp_YTDActivity SET EmployerID = " & vEmployerID & ", Year = " & vYear & ";"

```

'Update all records with Company name from EmployerData table, and match on EmployeeID to fill in Name, and MutualFundFamilies to fill in Mutual Fund Family

```

db.Execute "UPDATE ((Temp_YTDActivity LEFT JOIN EmployeeData ON Temp_YTDActivity.EmployeeID = EmployeeData.EmployeeID) LEFT JOIN EmployerData ON Temp_YTDActivity.EmployerID = EmployerData.EmployerID) LEFT JOIN MutualFundFamilies ON Temp_YTDActivity.MutualFundID = MutualFundFamilies.MutualFundID SET Temp_YTDActivity.Company = [EmployerData].[Company], Temp_YTDActivity.EmployeeName = [EmployeeData].[Last Name] & ', ' & [EmployeeData].[First Name] & ' ' & [EmployeeData].[Middle Initial], Temp_YTDActivity.SSNum = [EmployeeData].[SSNum], Temp_YTDActivity.MutualFund = [MutualFundFamilies].[Mutual Fund Family];"

```

End Sub

Public Sub pCreateTempYTDActivityByPortfolio(vYear, vEmployerID)

' Comments : Create Temp_YTDActivity table with all YTD activity by source and portfolio between a selected period plus one year and EmployerID

' Parameters : vYear -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

```
Dim db As Database
Dim vQueryStr As String
Set db = DBEEngine.Workspaces(0).Databases(0)
```

```
'Delete all records from Temp_YTDActivity table
db.Execute "Delete * from Temp_YTDActivity;"
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Contributions for selected year and next year, EmployerID, and source = 1
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Contributions ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=1)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Matchings for selected year and next year, EmployerID, and source = 2
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Matchings ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=2)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Loan payments for selected year and next year, EmployerID, and source = 3
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, LoanPayments ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=3)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Rollovers for selected year and next year, EmployerID, and source = 5
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Rollovers ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=5)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Transfers for selected year and next year, EmployerID, and source = 6
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Transfers ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=6)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```
'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Loan distributions for selected year and next year, EmployerID, and source = 7
```

```
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, LoanDistributions ) "
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND (Source=7)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr
```

```

'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Distributions for selected year and next year, EmployerID, and source =
9
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Distributions )"
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=9)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr

'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Other adjustments for selected year and next year, EmployerID, and
source = 10 or 11
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, OtherAdjustments )"
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=10 OR Source=11)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr

'Append records from AllocatedFunds for EmployeeID, PortfolioID and all Nonvested forfeitures for selected year and next year, EmployerID, and
source = 12
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, NonVestedForfeitures )"
vQueryStr = vQueryStr & "SELECT EmployeeID, PortfolioID, Sum(Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE ((EmployerID=" & vEmployerID & ") AND (Period Between " & vYear & " And " & vYear + 1 & ") AND
(Source=12)) "
vQueryStr = vQueryStr & "GROUP BY EmployeeID, PortfolioID;"
db.Execute vQueryStr

>Delete all records with EmployeeID is null or PortfolioID is null
vQueryStr = "DELETE Distinctrow * From Temp_YTDActivity Where IsNull(EmployeeID) or IsNull(PortfolioID);"
db.Execute vQueryStr

'Append summary records for all different types of amounts and set SummaryRecord to true
vQueryStr = "INSERT INTO Temp_YTDActivity ( EmployeeID, PortfolioID, Contributions, Matchings, LoanPayments, Rollovers, Transfers,
LoanDistributions, Distributions, OtherAdjustments, NonVestedForfeitures, SummaryRecord )"
vQueryStr = vQueryStr & "SELECT Temp_YTDActivity.EmployeeID, Temp_YTDActivity.PortfolioID, Sum(Temp_YTDActivity.Contributions)
AS SumOfContributions, Sum(Temp_YTDActivity.Matchings) AS SumOfMatchings, Sum(Temp_YTDActivity.LoanPayments) AS
SumOfLoanPayments, Sum(Temp_YTDActivity.Rollovers) AS SumOfRollovers, Sum(Temp_YTDActivity.Transfers) AS SumOfTransfers,
Sum(Temp_YTDActivity.LoanDistributions) AS SumOfLoanDistributions, Sum(Temp_YTDActivity.Distributions) AS SumOfDistributions,
Sum(Temp_YTDActivity.OtherAdjustments) AS SumOfOtherAdjustments, Sum(Temp_YTDActivity.NonVestedForfeitures) AS
SumOfNonVestedForfeitures, True AS Expr1 "
vQueryStr = vQueryStr & "FROM Temp_YTDActivity "
vQueryStr = vQueryStr & "GROUP BY Temp_YTDActivity.EmployeeID, Temp_YTDActivity.PortfolioID;"
db.Execute vQueryStr

>Delete all records that are not summary records
db.Execute "DELETE Distinctrow * FROM Temp_YTDActivity WHERE SummaryRecord = False;"

'Update all records with selected EmployerID and year
db.Execute "UPDATE Temp_YTDActivity SET EmployerID = " & vEmployerID & ", Year = " & vYear & ";"

'Update all records with Company name from EmployerData table, and match on EmployeeID to fill in Name, and Portfolios to fill in Portfolio
name
db.Execute "UPDATE ((Temp_YTDActivity LEFT JOIN EmployeeData ON Temp_YTDActivity.EmployeeID = EmployeeData.EmployeeID) LEFT
JOIN EmployerData ON Temp_YTDActivity.EmployerID = EmployerData.EmployerID) LEFT JOIN Portfolios ON Temp_YTDActivity.PortfolioID =
Portfolios.PortfolioID SET Temp_YTDActivity.Company = [EmployerData].[Company], Temp_YTDActivity.EmployeeName =
[EmployeeData].[Last Name] & ', ' & [EmployeeData].[First Name] & ' ' & [EmployeeData].[Middle Initial], Temp_YTDActivity.SSNNum =
EmployeeData.SSNNum, Temp_YTDActivity.MutualFund = Portfolios.PortfolioName;"

End Sub

Public Sub pCreateYTDSalaryGrid(vEmployerID, vYear)
' Comments : Deletes records and creates new records for the YTD salary for selected EmployerID and year
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

```

```

' Modified :
'
' _____

Dim db As Database
Dim vQueryStr As String
Set db = DBEngine.Workspaces(0).Databases(0)

'Delete all records in temporary table Temp_UpdateYTDSalary
db.Execute "Delete Distinctrow * From Temp_UpdateYTDSalary;"

'Create records for Temp_UpdateYTDSalary adding all monthly salary figures of each employee for selected year
' for the selected EmployerID and year from the ContributionData table
vQueryStr = "INSERT INTO Temp_UpdateYTDSalary ( EmployerID, EmployeeID, PriorYTD, EmployeeName, SSNum, UpdateYear ) "
vQueryStr = vQueryStr & "SELECT ContributionData.EmployerID, ContributionData.EmployeeID, Sum(ContributionData.Salary) AS
SumOfSalary, [Last Name] & ' ' & [First Name] & ' ' & [Middle Initial] AS EmployeeName, EmployeeData.SSNum, " & vYear & " AS Expr1 "
vQueryStr = vQueryStr & "FROM ContributionData INNER JOIN EmployeeData ON ContributionData.EmployeeID =
EmployeeData.EmployeeID "
vQueryStr = vQueryStr & "WHERE ((ContributionData.EmployerID = " & vEmployerID & ") And (ContributionData.Period Between " & vYear &
And " & vYear + 1 & ")) "
vQueryStr = vQueryStr & "GROUP BY ContributionData.EmployerID, ContributionData.EmployeeID, [Last Name] & ' ' & [First Name] & ' ' &
[Middle Initial], EmployeeData.SSNum;"
db.Execute vQueryStr

'Update the NewYTD field to equal to the calculated PriorYTD field
db.Execute "UPDATE Temp_UpdateYTDSalary SET Temp_UpdateYTDSalary.NewYTD = Temp_UpdateYTDSalary.PriorYTD;"

End Sub

Function periodtodate(CurrentPeriod, returntype)
' Comments : Function to format the curent period into different date formats based on return type
' CurrentPeriod looks like 1999.03 for March 1999
' Will return for returntype
' 1 March
' 2 March 31, 1999
' 3 March 1999
' 4 03/31/1999
'
' Parameters : CurrentPeriod -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim periodyear
Dim periodmonth As Integer
'5/22/2000 add RAH
Dim monthstring As String
Dim daystring As String
Dim yearstring As String

'If CurrentPeriod has a value other than zero
If CurrentPeriod << 0 Then
'Save the year from Currentperiod
periodyear = Int(CurrentPeriod)
'Save the month by subtracting the year leaving the decimal month multiplied by 100 for a whole number
periodmonth = (CurrentPeriod - periodyear) * 100
'Evaluate the month number
Select Case periodmonth
Case 1
'Save the month name
monthstring = "January"
'Save the number for the last day of the month
daystring = "31"
Case 2
'Save the month name

```

```

    monthstring = "February"
    'Save the number for the last day of the month
    daystring = "28"
Case 3
    'Save the month name
    monthstring = "March"
    'Save the number for the last day of the month
    daystring = "31"
Case 4
    'Save the month name
    monthstring = "April"
    'Save the number for the last day of the month
    daystring = "30"
Case 5
    'Save the month name
    monthstring = "May"
    'Save the number for the last day of the month
    daystring = "31"
Case 6
    'Save the month name
    monthstring = "June"
    'Save the number for the last day of the month
    daystring = "30"
Case 7
    'Save the month name
    monthstring = "July"
    'Save the number for the last day of the month
    daystring = "31"
Case 8
    'Save the month name
    monthstring = "August"
    'Save the number for the last day of the month
    daystring = "31"
Case 9
    'Save the month name
    monthstring = "September"
    'Save the number for the last day of the month
    daystring = "30"
Case 10
    'Save the month name
    monthstring = "October"
    'Save the number for the last day of the month
    daystring = "31"
Case 11
    'Save the month name
    monthstring = "November"
    'Save the number for the last day of the month
    daystring = "30"
Case 12
    'Save the month name
    monthstring = "December"
    'Save the number for the last day of the month
    daystring = "31"
End Select
'Save year converted to string
yearstring = CStr(periodyear)
'Evaluate the format type
Select Case returntype
Case 1
    'Return month name
    periodtodate = monthstring
Case 2
    'Return month name last day of month, and year
    periodtodate = monthstring & " " & daystring & ", " & yearstring
Case 3
    'Return month name and year
    periodtodate = monthstring & " " & yearstring
Case 4
    'Return month/last day of month/year
    periodtodate = CVDate(CStr(periodmonth) & "/" & daystring & "/" & CStr(periodyear))

```

```
End Select
Else
    'No date supplied, return None
    periodtodate = "None"
End If
```

End Function

Function perminus(ByVal Period)

```
' Comments : This routine will subtract .01 from mm to get the prior month. _
            It will check to see if this spans back a year.
' Parameters : Period - in the form of a number representing yyyy.mm
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

'Subtract 1 from the month
Period = Period - 0.01

'If the remainder is 0 then the date will be December of the previous year (yyyy-1.12)
If ((cfix2(Period - Int(Period)) * 100) = 0) Then
    Period = Period - 0.88
End If

'Return the year and month as a number represented by yyyy.mm
perminus = cfix2(Period)
```

End Function

Function perplus(ByVal Period)

```
' Comments : This routine will add .01 to mm to get the next month. _
            It will check to see if this spans to the next year.
' Parameters : Period - in the form of a number representing yyyy.mm
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

'Add 1 to the month
Period = Period + 0.01

'If the remainder is 13 then the date will be January of the next year (yyyy+1.01)
If ((cfix2(Period - Int(Period)) * 100) = 13) Then
    Period = Period + 0.88
End If

'Return the year and month as a number represented by yyyy.mm
perplus = cfix2(Period)
```

End Function

Public Sub pLoadTempBatchData(vBatchID, vEmployerID)

```
' Comments : Load temporary tables Temp_BatchData and Temp_BatchMF with records of employees for selected
            BatchID and EmployerID
' Parameters : vBatchID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

Dim db As Database
Dim vQueryStr As String
Set db = DBEEngine.Workspaces(0).Databases(0)
```

```
'Clear all records from Temp_BatchData table and Temp_BatchMF table
pClearTempBatchData
```

```
'Append records from AllocatedFunds for EmployeeID and total of all amounts for selected BatchID
vQueryStr = "INSERT INTO Temp_BatchData ( EmployeeID, BatchAmount ) "
vQueryStr = vQueryStr & "SELECT AllocatedFunds.EmployeeID, Sum(AllocatedFunds.Amount) AS SumOfAmount "
vQueryStr = vQueryStr & "FROM AllocatedFunds "
vQueryStr = vQueryStr & "WHERE (((AllocatedFunds.BatchID) = " & vBatchID & ") "
vQueryStr = vQueryStr & "GROUP BY AllocatedFunds.EmployeeID;"
db.Execute vQueryStr
```

```
'Append records from EmployeeData for EmployeeID who match the selected EmployerID
vQueryStr = "INSERT INTO Temp_BatchData ( EmployeeID ) "
vQueryStr = vQueryStr & "SELECT EmployeeData.EmployeeID "
vQueryStr = vQueryStr & "FROM EmployeeData "
vQueryStr = vQueryStr & "WHERE (((EmployeeData.EmployerID) = " & vEmployerID & ");"
db.Execute vQueryStr
```

```
'Append summary records for totals, employee name, social security number, and summary record
vQueryStr = "INSERT INTO Temp_BatchData ( EmployeeID, BatchAmount, EmployeeName, SSNum, SummaryRecord ) "
vQueryStr = vQueryStr & "SELECT Temp_BatchData.EmployeeID, Sum(Temp_BatchData.BatchAmount) AS SumOfBatchAmount, [Last Name]
& ', ' & [First Name] & ' ' & [Middle Initial] AS Expr1, EmployeeData.SSNum, True AS Expr2 "
vQueryStr = vQueryStr & "FROM Temp_BatchData INNER JOIN EmployeeData ON Temp_BatchData.EmployeeID = EmployeeData.EmployeeID
"
vQueryStr = vQueryStr & "GROUP BY Temp_BatchData.EmployeeID, [Last Name] & ', ' & [First Name] & ' ' & [Middle Initial],
EmployeeData.SSNum;"
db.Execute vQueryStr
```

```
'Delete all records that are not summary record
db.Execute "Delete Distinctrow * From Temp_BatchData Where SummaryRecord = False;"
```

```
'Append employee records to Temp_BatchMF table with PortfolioID, Allocation, and Account number
vQueryStr = "INSERT INTO Temp_BatchMF ( EmployeeID, PortfolioID, Allocation, AccountNum ) "
vQueryStr = vQueryStr & "SELECT EmployeeMF.EmployeeID, EmployeeMF.PortfolioID, EmployeeMF.Allocation, EmployeeMF.AccountNum "
vQueryStr = vQueryStr & "FROM EmployeeMF "
vQueryStr = vQueryStr & "WHERE (((EmployeeMF.EmployerID)= " & vEmployerID & ");"
db.Execute vQueryStr
```

End Sub

Public Sub pSaveRecord()

```
' Comments : Procedure to call a function to save the current record
' Parameters : -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
Dim vReturnValue As Variant
'Call function for generic save current record
vReturnValue = fSaveRecord()
```

End Sub

Public Sub SetEstimatedContrib(vEmployerID, vYear)

```
' Comments : Calculates the yearly estimated contributions based on past salary contributions
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____
```

```
Dim ws As Workspace
Dim db As Database
Dim rs
Dim rs1
```

```

Dim rs2 As Recordset
Dim vNumPeriods As Integer
Dim vAvgContrib As Currency

Set ws = DBEEngine.Workspaces(0)
Set db = ws.Databases(0)

'Set mouse to hourglass
DoCmd.Hourglass True

'Set transaction processing to start
ws.BeginTrans
'Get all employee records for selected EmployerID from Temp_ADPTTest table
Set rs = db.OpenRecordset("Select Distinctrow * from Temp_ADPTTest Where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get the average contribution for each employee for selected EmployerID and year from ContributionData table
Set rs1 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Avg(ContribAmt) AS AverageContrib FROM ContributionData WHERE EmployerID = " & vEmployerID & " AND Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)
'Get the lowest period for each employee for selected EmployerID and year from ContributionData table
Set rs2 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Min(Period) as FirstPeriod from ContributionData Where EmployerID = " & vEmployerID & " and Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)

Dim db As Database
Dim rs
Dim rs1
Dim rs2 As Recordset
Dim vNumPeriods As Integer
Dim vAvgMatch As Currency

Set ws = DBEEngine.Workspaces(0)
Set db = ws.Databases(0)

'Set mouse to hourglass
DoCmd.Hourglass True

'Set transaction processing to start
ws.BeginTrans
'Get all employee records for selected EmployerID from Temp_ADPTTest table
Set rs = db.OpenRecordset("Select Distinctrow * from Temp_ADPTTest Where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get the average match for each employee for selected EmployerID and year from ContributionData table
Set rs1 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Avg(MatchAmt) AS AverageMatch FROM ContributionData WHERE EmployerID = " & vEmployerID & " AND Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)
'Get the lowest period for each employee for selected EmployerID and year from ContributionData table
Set rs2 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Min(Period) as FirstPeriod from ContributionData Where EmployerID = " & vEmployerID & " and Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)

'Read each employee record
Do Until rs.EOF
'Find average match record for current EmployeeID
rs1.FindFirst "EmployeeID = " & rs&EmployeeID
'If average match record does not match
If rs1.NoMatch Then
'Prepare record for update
rs.Edit
'Set EstMatch to zero
rs&EstMatch = 0
'Write record
rs.Update
'If average match record does match
Else
'Find period record for current EmployeeID
rs2.FindFirst "EmployeeID = " & rs&EmployeeID
'Save AvgMatch rounded to two places
vAvgMatch = cfix2(rs1&AverageMatch)
'Save the number of periods by converting to decimal and subtracting from 12 minus 1 to be inclusive
vNumPeriods = 12 - (cfix2((rs2&FirstPeriod - Int(rs2&FirstPeriod)) * 100) - 1)
'Prepare record for update
rs.Edit

```

```

        'Set the EstMatch to average match times number of periods rounded to two places
        rs&EstMatch = cfix2(vAvgMatch * vNumPeriods)
        'Write record
        rs.Update
    End If
    'Move to next employee record
    rs.MoveNext
'Go back to top of Exit for next employee
Exit
'Close employee recordset
rs.Close
'Close average match recordset
rs1.Close
'Close period recordset
rs2.Close

'Commit transaction processing
ws.CommitTrans
'Set mouse back to pointer
DoCmd.Hourglass False

End Sub

Sub SetEstimatedSalary(vEmployerID, vYear)
' Comments : Calculates the yearly estimated salary based on past salary history.
' Parameters : vEmployerID -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

' Calculates the yearly estimated salary based on past salary history.

Dim ws As Workspace
Dim db As Database
Dim rs
Dim rs1
Dim rs2 As Recordset
Dim vNumPeriods As Integer
Dim vAvgSalary As Currency

Set ws = DBEngine.Workspaces(0)
Set db = ws.Databases(0)

'Set mouse to hourglass
DoCmd.Hourglass True

'Set transaction processing to start
ws.BeginTrans
'Get all employee records for selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * from EmployeeData Where EmployerID = " & vEmployerID & ";", dbOpenDynaset)
'Get the average salary for each employee for selected EmployerID and year from ContributionData table
Set rs1 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Avg(Salary) AS AverageSalary FROM ContributionData WHERE
EmployerID = " & vEmployerID & " AND Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)
'Get the lowest period for each employee for selected EmployerID and year from ContributionData table
Set rs2 = db.OpenRecordset("SELECT DISTINCTROW EmployeeID, Min(Period) as FirstPeriod from ContributionData Where EmployerID = " &
vEmployerID & " and Int(Period) = " & vYear & " GROUP BY EmployeeID;", dbOpenSnapshot)

'Read each employee record
Do Until rs.EOF
'Find salary record for current EmployeeID
rs1.FindFirst "EmployeeID = " & rs&EmployeeID
'If a salary record does not match
If rs1.NoMatch Then
'Prepare record for update
rs.Edit
'Set EstSalary to 0

```

```

rs&EstSalary = 0
'Write record
rs.Update
'If a salary record does match
Else
'Find period record for current EmployeeID
rs2.FindFirst "EmployeeID = " & rs&EmployeeID
'Save average salary rounded to two places
vAvgSalary = cfix2(rs1&AverageSalary)
'Save the number of periods by converting to decimal and subtracting from 12 minus 1 to be inclusive
vNumPeriods = 12 - (cfix2((rs2&FirstPeriod - Int(rs2&FirstPeriod)) * 100) - 1)
'Prepare record for update
rs.Edit
'Set the EstSalary to average salary times number of periods rounded to two places
rs&EstSalary = cfix2(vAvgSalary * vNumPeriods)
'Write record
rs.Update
End If
'Move to next employee record
rs.MoveNext
'Go back to top of Exit for next employee
Exit
'Close employee recordset
rs.Close
'Close salary recordset
rs1.Close
'Close period recordset
rs2.Close

'Commit transaction processing
ws.CommitTrans
'Set mouse back to pointer
DoCmd.Hourglass False

```

End Sub

Sub SetHCandKey(employeridnum, YearNum, vHCBasis, vTop20Switch)

' Comments : Sets Highly Compensated and Key employees s for 5500 Summary Report

' Parameters : employeridnum

' vHCBasis -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

' _____

' Sets Highly Compensated and Key employees for 5500 Summary Report

Dim ws As Workspace

Dim db As Database

Dim rs As Recordset

Dim top20 As Long

Dim ownerlimit As Currency ' Limit for 1% owner

Dim employeelimit As Currency ' Limit for employee

Dim officerlimit As Currency ' Limit for officer to be key employee (2002)

Dim top20salary As Currency ' The bottom salary of the top 20% group

Set ws = DBEngine.Workspaces(0)

Set db = ws.Databases(0)

'Create a recordset using T_ComplianceTesting for selected year

Set rs = db.OpenRecordset("Select Distinctrow * From T_ComplianceTesting Where TestingYear = " & YearNum & ";", dbOpenSnapshot)

'Save OwnerSalary

ownerlimit = rs&OwnerSalary

'Save EmployeeSalary

employeelimit = rs&EmployeeSalary

'Save OfficerSalary

officerlimit = rs&OfficerSalary

```

'Changed RAH 1/18/2002 because limits are current year even though basis is Last Year
' If vHCBasis value is 1
' If vHCBasis = 1 Then
'     'Create a recordset using T_ComplianceTesting for selected year - 1
'     Set rs = db.OpenRecordset("Select Distinctrow * From T_ComplianceTesting Where TestingYear = " & YearNum - 1 & ";",
dbOpenSnapshot)
'     'If any records are found save EmployeeLimit from EmployeeSalary in the first record
'     If rs.RecordCount < 0 Then employeelimit = rs&EmployeeSalary
'     End If

'Close the recordset
rs.Close

'Set mouse to hourglass
DoCmd.Hourglass True

'Begin transaction processing
ws.BeginTrans
'If value of vTop20Switch is true
If vTop20Switch = True Then
'Evaluate vHCBasis
Select Case vHCBasis
'Prior Year Salary
Case 1
'Count the number of eligible employees from EmployeeData table based on selected EmployerID
Set rs = db.OpenRecordset("SELECT Count(EmployeeID) as numemployees FROM EmployeeData WHERE EmployerID = " &
employeridnum & " AND Eligible = True;"; dbOpenSnapshot)
'Save the count of employees multiplied by .20 as a whole number rounded up
top20 = CLng(rs&NumEmployees * 0.2)
'If Top20 equals 0 change it to 1
If top20 = 0 Then top20 = 1
'Create a recordset from EmployeeData of eligible employees sorted by PriorYearSalary in descending order fro selected EmployerID
Set rs = db.OpenRecordset("SELECT DISTINCTROW * FROM EmployeeData WHERE EmployerID = " & employeridnum & " AND
Eligible = True ORDER BY PriorYearSalary DESC;"; dbOpenSnapshot)
'Move to record number that is 1 less than the value of Top20
rs.Move (top20 - 1)
'Save the PriorYearSalary from that record
top20salary = rs&PriorYearSalary
'Estimated Salary
Case 2
'Count the number of eligible employees from EmployeeData table based on selected EmployerID
Set rs = db.OpenRecordset("SELECT Count(EmployeeID) as numemployees FROM EmployeeData WHERE EmployerID = " &
employeridnum & " AND Eligible = True;"; dbOpenSnapshot)
'Save the count of employees multiplied by .20 as a whole number rounded up
top20 = CLng(rs&NumEmployees * 0.2)
'If Top20 equals 0 change it to 1
If top20 = 0 Then top20 = 1
'Create a recordset from EmployeeData of eligible employees sorted by EstSalary in descending order fro selected EmployerID
Set rs = db.OpenRecordset("SELECT DISTINCTROW * FROM EmployeeData WHERE EmployerID = " & employeridnum & " AND
Eligible = True ORDER BY EstSalary DESC;"; dbOpenSnapshot)
'Move to record number that is 1 less than the value of Top20
rs.Move (top20 - 1)
'Save the EstSalary from that record
top20salary = rs&EstSalary
End Select
End If

'Create recordset of EmployeeData of eligible employees for selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * from EmployeeData where EmployerID = " & employeridnum & " and Eligible = True;";
dbOpenDynaset)

'Read through all EmployeeData records
Do Until rs.EOF
'Prepare for record update
rs.Edit
'Set KeyEmployee to false
rs&KeyEmployee = False
'Set HCEmployee to false

```

```

rs&HCEmployee = False

'If employee is officer then set KeyEmployee to true
If rs&Officer = True Then
    If YearNum = 2002 Then
        If rs&EstSalary <= officerlimit Then
            rs&KeyEmployee = True
        End If
    Else
        rs&KeyEmployee = True
    End If
End If

'If employee is 5p owner
If rs&[5P Owner] = True Then
    'Set KeyEmployee to true
    rs&KeyEmployee = True
    'Set HCEmployee to true
    rs&HCEmployee = True
End If
'If employee is 1p owner and estimated salary equals or is greater than OwnerLimit set KeyEmployee to true
If (rs&[1P Owner] = True) And (rs&EstSalary <= ownerlimit) Then rs&KeyEmployee = True
'Evaluate vHCBasis
Select Case vHCBasis
    'Prior Year Salary
    Case 1
        'If vTop20Switch is true
        If vTop20Switch = True Then
            'If Prior Year Salary is greater than employee limit and Prior Year Salary equal or greater then Top20salary then set HCEmployee to true
            If (rs&PriorYearSalary < employeelimit) And (rs&PriorYearSalary <= top20salary) Then rs&HCEmployee = True
        Else
            'Top20Switch is not true
            'If Prior Year Salary greater than Employee Limit then set HCEmployee to true
            If rs&PriorYearSalary < employeelimit Then rs&HCEmployee = True
        End If
    'Estimated Salary
    Case 2
        'If vTop20Switch is true
        If vTop20Switch = True Then
            'If Estimated Salary is greater than employee limit and Estimated Salary equal or greater then Top20salary then set HCEmployee to true
            If (rs&EstSalary < employeelimit) And (rs&EstSalary <= top20salary) Then rs&HCEmployee = True
        Else
            'Top20Switch is not true
            'If Estimated Salary greater than Employee Limit then set HCEmployee to true
            If rs&EstSalary < employeelimit Then rs&HCEmployee = True
        End If
    End Select
'Write record
rs.Update
'Move to next EmployeeData record
rs.MoveNext
'Go back to Exit to process next EmployeeData record
Exit
'Close EmployeeData recordset
rs.Close

'Commit transaction processing
ws.CommitTrans
'Set mouse to pointer
DoCmd.Hourglass False

```

End Sub

Sub SetHCandKeyADP()

' Comments : Sets Highly Compensated and Key employees s for ADP, M Test, and Top Heavy Test.

' Parameters : -

' Returns : -

' Created :

' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC

' Modified :

'

' _____

' Sets Highly Compensated and Key employees for ADP, M Test, and Top Heavy Test.

Dim ws As Workspace

Dim db As Database

Dim rs

Dim rs1 As Recordset

Dim vEmployerID

Dim vTop20 As Long

Dim vYear

Dim vHCBasis As Integer

Dim vTop20Switch As Boolean

Dim vOwnerLimit As Currency

Dim vEmployeeLimit As Currency

Dim vOfficerLimit As Currency

Dim vTop20salary As Currency

Set ws = DBEngine.Workspaces(0)

Set db = ws.Databases(0)

'Create a recordset using Temp_ADPTTest table

Set rs = db.OpenRecordset("Temp_ADPTTest", dbOpenSnapshot)

'Save EmployerID

vEmployerID = rs&EmployerID

'Save year

vYear = rs&Year

'Save HCBasis

vHCBasis = rs&HCBasis

'Save Top20Switch

vTop20Switch = rs&Top20Switch

'Create a recordset using T_ComplianceTesting for same year as vYear

Set rs = db.OpenRecordset("Select Distinctrow * From T_ComplianceTesting Where TestingYear = " & vYear & ";", dbOpenSnapshot)

'Save OwnerSalary

vOwnerLimit = rs&OwnerSalary

'Save EmployeeSalary

vEmployeeLimit = rs&EmployeeSalary

'Save OfficerSalary

vOfficerLimit = rs&OfficerSalary

'Changed RAH 1/18/2002 because limits are current year even though basis is Last Year

' If vHCBasis value is 1

' If vHCBasis = 1 Then

' Create a recordset using T_ComplianceTesting for vYear - 1

' Set rs = db.OpenRecordset("Select Distinctrow * From T_ComplianceTesting Where TestingYear = " & vYear - 1 & ";", dbOpenSnapshot)

' If any records are found save EmployeeLimit from EmployeeSalary in the first record

' If rs.RecordCount < 0 Then vEmployeeLimit = rs&EmployeeSalary

' End If

'Close the recordset

rs.Close

'Set mouse to hourglass

DoCmd.Hourglass True

'Begin transaction processing

ws.BeginTrans

'If value of vTop20Switch is true

If vTop20Switch = True Then

'Evaluate vHCBasis

Select Case vHCBasis

'Prior Year Salary

Case 1

'Count the number of eligible employees from Temp_ADPTTest table

Set rs = db.OpenRecordset("SELECT Count(EmployeeID) as NumEmployees FROM Temp_ADPTTest WHERE Eligible = True;", dbOpenSnapshot)

```

        'Save the count of employees multiplied by .20 as a whole number rounded up
vTop20 = CLng(rs&NumEmployees * 0.2)
        'If vTop20 equals 0 change it to 1
If vTop20 = 0 Then vTop20 = 1
        'Create a recordset from Temp_ADPTTest of eligible employees sorted by PriorYearSalary in descending order
Set rs = db.OpenRecordset("SELECT DISTINCTROW * FROM Temp_ADPTTest WHERE Eligible = True ORDER BY PriorYearSalary
DESC;"; dbOpenSnapshot)
        'Move to record number that is 1 less than the value of vTop20
rs.Move (vTop20 - 1)
        'Save the PriorYearSalary from that record
vTop20salary = rs&PriorYearSalary
        'Estimated Salary
Case 2
        'Count the number of eligible employees from Temp_ADPTTest table
Set rs = db.OpenRecordset("SELECT Count(EmployeeID) as NumEmployees FROM Temp_ADPTTest WHERE Eligible = True;";
dbOpenSnapshot)
        'Save the count of employees multiplied by .20 as a whole number rounded up
vTop20 = CLng(rs&NumEmployees * 0.2)
        'If vTop20 equals 0 change it to 1
If vTop20 = 0 Then vTop20 = 1
        'Create a recordset from Temp_ADPTTest of eligible employees sorted by EstSalary in descending order
Set rs = db.OpenRecordset("SELECT DISTINCTROW * FROM Temp_ADPTTest WHERE Eligible = True ORDER BY EstSalary DESC;";
dbOpenSnapshot)
        'Move to record number that is 1 less than the value of vTop20
rs.Move (vTop20 - 1)
        'Save the EstSalary from that record
vTop20salary = rs&EstSalary
    End Select
End If

'Create recordset of Temp_ADPTTest
Set rs = db.OpenRecordset("Select Distinctrow * From Temp_ADPTTest Order By EmployeeID;"; dbOpenDynaset)
'Create recordset of EmployeeData for selected EmployerID
Set rs1 = db.OpenRecordset("Select Distinctrow * From EmployeeData Where EmployerID = " & vEmployerID & " Order By EmployeeID;";
dbOpenSnapshot)

'Read through all Temp_ADPTTest records
Do Until rs.EOF
    'Prepare for record update
rs.Edit
    'Set KeyEmployee to false
rs&KeyEmployee = False
    'Set HCEmployee to false
rs&HCEmployee = False
    'If employee is eligible
If rs&Eligible = True Then
        'Get Employee record that matches Temp_ADPTTest mployeeID
rs1.FindFirst "EmployeeID = " & rs&EmployeeID

        'If employee is officer then set KeyEmployee to true
If rs1&Officer = True Then
            If vYear = 2002 Then
                If rs&EstSalary <= vOfficerLimit Then
                    rs&KeyEmployee = True
                End If
            Else
                rs&KeyEmployee = True
            End If
        End If

        'If employee is 5p owner
If rs1&[5P Owner] = True Then
            'Set KeyEmployee to true
rs&KeyEmployee = True
            'Set HCEmployee to true
rs&HCEmployee = True
        End If

        'If employee is 1p owner and salary equals or is greater than vOwnerLimit set KeyEmployee to true
If (rs1&[1P Owner] = True) And (rs&EstSalary <= vOwnerLimit) Then rs&KeyEmployee = True
        'Evaluate vHCBasis

```

```

Select Case vHCBasis
    'Prior Year Salary
    Case 1
        'If vTop20Switch is true
        If vTop20Switch = True Then
            'If Prior Year Salary is greater than employee limit and Prior Year Salary equal or greater then vTop20salary then set HCEmployee to
true
                If (rs&PriorYearSalary < vEmployeeLimit) And (rs&PriorYearSalary <= vTop20salary) Then rs&HCEmployee = True
            Else
                'vTop20Switch is not true
                'If Prior Year Salary greater than Employee Limit then set HCEmployee to true
                If rs&PriorYearSalary < vEmployeeLimit Then rs&HCEmployee = True
            End If
        'Estimated Salary
        Case 2
            'If vTop20Switch is true
            If vTop20Switch = True Then
                'If Estimated Salary is greater than employee limit and Estimated Salary equal or greater then vTop20salary then set HCEmployee to
true
                    If (rs&EstSalary < vEmployeeLimit) And (rs&EstSalary <= vTop20salary) Then rs&HCEmployee = True
                Else
                    'vTop20Switch is not true
                    'If Estimated Salary greater than Employee Limit then set HCEmployee to true
                    If rs&EstSalary < vEmployeeLimit Then rs&HCEmployee = True
                End If
            End Select
        End If
    'Write record
    rs.Update
    'Move to next Temp_ADPTTest record
    rs.MoveNext
    'Go back to Exit to process next Temp_ADPTTest record
    Exit
    'Close Temp_ADPTTest recordset
    rs.Close
    'Close EmployeeData recordset
    rs1.Close

    'Commit transaction processing
    ws.CommitTrans
    'Set mouse to pointer
    DoCmd.Hourglass False

End Sub

Public Sub SetPriorYearSalary(employeridnum, YearNum)
    ' Comments : Calculates and posts the total prior year salary for compliance testing.
    ' Parameters : employeridnum -
    ' Returns : -
    ' Created :
    ' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
    ' Modified :
    '
    ' _____

    ' Calculates and posts the total prior year salary for compliance testing.

    Dim db As Database
    Dim rs As Recordset
    Dim rs1 As Recordset

    Set db = DBEngine.Workspaces(0).Databases(0)
    'Get all employee records for selected EmployerID
    Set rs = db.OpenRecordset("Select Distinctrow * from EmployeeData where EmployerID = " & employeridnum & ";", dbOpenDynaset)
    'Get the total of Salary from query SumOfYearlySalaryQ for all employees for selected employer and year
    Set rs1 = db.OpenRecordset("Select Distinctrow * from SumOfYearlySalaryQ where EmployerID = " & employeridnum & " and Year = " &
YearNum & ";", dbOpenSnapshot)
    'Read all employee records
    Do Until rs.EOF

```

```

'Find the salary record that matches the current EmployeeID
rs1.FindFirst "EmployeeID = " & rs&EmployeeID
'prepare the record for update
rs.Edit
'If a salary record matches
If Not rs1.NoMatch Then
    'Match, so set the field PriorYearSalary to total salary
    rs&PriorYearSalary = rs1&SumOfSalary
Else
    'No match, so set the field PriorYearSalary to zero
    rs&PriorYearSalary = 0
End If
'Write record
rs.Update
'Move to next employee record
rs.MoveNext
'Go back to top of Exit for next employee
Exit
'Close employee recordset
rs.Close
'Close total salary recordset
rs1.Close

```

End Sub

```

Public Sub SetYTDCCompensation(employeridnum, YearNum)
' Comments : Calculates and posts the total year to date compensation.
' Parameters : employeridnum -
' Returns : -
' Created :
' Documented : April, 2000 by Alison J. Balter, Marina Consulting Group LLC
' Modified :
'
' _____

```

```

' Calculates and posts the total year to date compensation.

```

```

Dim db As Database
Dim rs As Recordset
Dim rs1 As Recordset

```

```

Set db = DBEngine.Workspaces(0).Databases(0)
'Get all employee records for selected EmployerID
Set rs = db.OpenRecordset("Select Distinctrow * from EmployeeData where EmployerID = " & employeridnum & ";", dbOpenDynaset)
'Read all employee records
Do Until rs.EOF
    'Get the total of Salary from ContributionData for current employee for selected year
    Set rs1 = db.OpenRecordset("Select Sum(Salary) as totsalary from ContributionData where EmployeeID = " & rs&EmployeeID & " and
CInt(Period) = " & YearNum & ";", dbOpenSnapshot)
    'Prepare record for update
    rs.Edit
    'Set the field YTDCCompensation to the total salary rounded to two places
    rs&YTDCCompensation = cfix2(nullto0(rs1&TotSalary))
    'Write record
    rs.Update
'Close recordset of salary records
rs1.Close
'Move to next employee record
rs.MoveNext
'Go back to top of Exit for next employee
Exit
'Close recordset of employees
rs.Close

```

End Sub1