Open UserForm, Insert Listbox, Labels, Textboxes and Command Button,

Change name of List box (MyDataList), command Buttons (Like CmdSum, CmdMax, CmdMin, CmdAvg), text boxes from Properties.

I am using "Data" sheet which have 500 Data.

Name of List Box is : DataListBox (Select Listbox and Change the name from Properties)

Private Sub DataListDisplay()

```
Dim ws As Worksheet
Set ws = ThisWorkbook.Worksheets("Data")

Dim lastRow As Long
lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

Dim rng As Range
Set rng = ws.Range("A1:L" & lastRow)
DataListBox.Clear
DataListBox.List = rng.value

TotRecord.Caption = "Total Records: " & lastRow - 1

Dim TotalY, TotalN, TotalYN
Dim i As Integer
Dim YesNo As String
YesNo = "Y"

For i = 2 To lastRow
    If ws.Cells(i, 8).value = YesNo Then
     TotalY = TotalY + ws.Cells(i, "D").value
    Else
```

```
        TotalN = TotalN + ws.Cells(i, "D").value
         End If
        TotalYN = TotalYN + ws.Cells(i, "D").value
        Next i

    'AmtRecd.Caption = "Rs. " & Format(TotalY, "###,##0.00")
   ' AmtDues.Caption = "Rs. " & Format(TotalN, "###,##0.00")
    TotAmt.Caption = "Rs. " & Format(TotalYN, "###,##0.00")

End Sub
```

Definition of the Excel VBA command :

Dim ws As Worksheet Set ws = ThisWorkbook.Worksheets("Data")

Dim lastRow As Long lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

Dim rngAmount As Range Set rngAmount = ws.Range("D2:D" & lastRow)

The above code snippet demonstrates how to work with worksheets, rows, and ranges in Excel VBA. Here's a breakdown of each line:

Dim ws As Worksheet:

This line declares a variable named ws with the data type Worksheet.

It creates a reference to a specific worksheet within the workbook.

Set ws = ThisWorkbook.Worksheets("Data"):

This code defines a subroutine called DataListDisplay which populates a list box (presumably named DataListBox) with data from an Excel worksheet. It also calculates and displays total amounts based on certain conditions. Here's a breakdown of the code:

DataListDisplay Subroutine:

The subroutine is triggered to initialize and populate the list box with data.

Worksheet and Range Setup:

The worksheet named "Data" in the current workbook is assigned to the ws variable.

The code determines the last row with data in column A of the worksheet using the .End(xlUp) method.

Populating the List Box:

A range rng is set to cover the columns A to L and rows 1 to lastRow (including headers).

The data from the range rng is assigned to the .List property of the DataListBox, populating it.

Total Record Count:

The caption of the TotRecord label is set to display the total number of records (excluding the header row).

Calculating Totals:

Variables TotalY, TotalN, and TotalYN are declared to keep track of calculated totals.

A loop iterates through rows (from 2 to lastRow) of the worksheet data.

If the value in column 8 (presumably indicating a "Yes" or "No") is "Y", the corresponding amount from column "D" is added to TotalY.

If the value is not "Y", the amount is added to TotalN.

The amount is always added to TotalYN.

Caption Updates:

The captions of AmtRecd, AmtDues, and TotAmt labels are commented out, but the last one is used to display the total amount (TotalYN) formatted as currency.

In summary, this subroutine populates the list box with data, calculates and displays total amounts based on certain conditions, and updates label captions to show record counts and total amounts.

Command Button Name is CmdMax

```
Private Sub CmdMax_Click()

    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Data")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    Dim rngAmount As Range
    Set rngAmount = ws.Range("D2:D" & lastRow) ' Assuming Amount column is column D

    Dim SumMax As Double
    SumMax = WorksheetFunction.Max(rngAmount)

    LblMax.Caption = "Maximum: Rs. " & Format(SumMax, "###,##0.00")

End Sub
```

This code defines a subroutine named CmdMax_Click which is executed when the "Max" button (presumably named CmdMax) is clicked. The subroutine calculates and displays the maximum value from a range of data and updates a label (LblMax) with the result. Here's a breakdown of the code:

CmdMax_Click Subroutine:

This subroutine is triggered when the "Max" button is clicked.

Worksheet and Range Setup:

The worksheet named "Data" in the current workbook is assigned to the ws variable.

The code determines the last row with data in column A of the worksheet using the .End(xlUp) method.

Range Setup for Calculation:

A range rngAmount is set to cover column D from row 2 to the last populated row (lastRow).

Calculating Maximum Value:

The WorksheetFunction.Max function is used to calculate the maximum value from the rngAmount range.

The calculated maximum value is stored in the SumMax variable.

Updating Label Caption:

The caption of the LblMax label is updated to display the calculated maximum value (SumMax) formatted as currency.

In summary, this subroutine calculates the maximum value from a specified range of data in the "Data" worksheet, and then updates a label to display the calculated maximum value.

Command Button Name is CmdMin

Private Sub CmdMin_Click()

```
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Data")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    Dim rngAmount As Range
    Set rngAmount = ws.Range("D2:D" & lastRow) ' Assuming Amount column is column D

    Dim SumMin As Double
    SumMin = WorksheetFunction.Min(rngAmount)

    LblMin.Caption = "Minimum: Rs. " & Format(SumMin, "###,##0.00")

End Sub
```

CmdMin_Click Subroutine:

This subroutine is triggered when the "Min" button is clicked.

Worksheet and Range Setup:

The worksheet named "Data" in the current workbook is assigned to the ws variable.

The code determines the last row with data in column A of the worksheet using the .End(xlUp) method.

Range Setup for Calculation:

A range rngAmount is set to cover column D from row 2 to the last populated row (lastRow).

Calculating Minimum Value:

The WorksheetFunction.Min function calculates the minimum value from the rngAmount range.

The calculated minimum value is stored in the SumMin variable.

Updating Label Caption:

The caption of the LblMin label is updated to display the calculated minimum value (SumMin) formatted as currency.

In summary, this subroutine calculates the minimum value from a specified range of data in the "Data" worksheet and then updates a label to display the calculated minimum value.

Command Button name is CmdSum

```
Private Sub CmdSum_Click()

    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Data")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    Dim rngAmount As Range
    Set rngAmount = ws.Range("D2:D" & lastRow) ' Assuming Amount column is column D
```

```
    Dim sumAmount As Double
    sumAmount = WorksheetFunction.Sum(rngAmount)

    LblSum.Caption = "Sum: Rs. " & Format(sumAmount, "###,##0.00")

End Sub
```

This code defines a subroutine named CmdSum_Click that is triggered when the "Sum" button (likely named CmdSum) is clicked. The subroutine calculates and displays the sum of values from a specified range of data, updating a label (LblSum) with the result. Here's a breakdown of the code:

CmdSum_Click Subroutine:

This subroutine is executed when the "Sum" button is clicked.

Worksheet and Range Setup:

The worksheet named "Data" in the current workbook is assigned to the ws variable.

The code determines the last row with data in column A of the worksheet using the .End(xlUp) method.

Range Setup for Calculation:

A range rngAmount is set to cover column D from row 2 to the last populated row (lastRow).

Calculating Sum:

The WorksheetFunction.Sum function calculates the sum of values from the rngAmount range.

The calculated sum is stored in the sumAmount variable.

Updating Label Caption:

The caption of the LblSum label is updated to display the calculated sum (sumAmount) formatted as currency.

In summary, this subroutine calculates the sum of values from a specified range of data in the "Data" worksheet and updates a label to display the calculated sum.

Command Button Name is CmdAverage

```
Private Sub CmdAverage_Click()

    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Data")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    Dim rngAmount As Range
    Set rngAmount = ws.Range("D2:D" & lastRow) ' Assuming Amount column is column D

    Dim sumAve As Double
    sumAve = WorksheetFunction.Average(rngAmount)

    LblAve.Caption = "Average: Rs. " & Format(sumAve, "###,##0") & "/-"

End Sub
```

This code calculates the average of values in a specified range ("D2:D" & lastRow) in the "Data" worksheet and displays the result in a label (LblAve) on the user form. Here's a breakdown of the code:

Worksheet Setup:

The code specifies the "Data" worksheet to work with using the ws variable.

Last Row Calculation:

The code determines the last used row in column A of the "Data" worksheet (lastRow). This is often used to establish the range of data.

Setting Range:

A range (rngAmount) is defined to represent column D ("Amount") from row 2 to the last row.

Calculating Average:

The WorksheetFunction.Average function is used to calculate the average of the values within the specified range (rngAmount).

The calculated average is stored in the sumAve variable.

Displaying Average:

The LblAve.Caption is updated with the calculated average value.

The Format function is used to format the average as a currency value with commas and a "/-" symbol at the end.

The purpose of this code is to provide functionality to calculate the average of values in a specific column ("Amount") of the "Data" worksheet and display the result in a user interface element (LblAve).

Command Button Name is CmsFind

```
Private Sub CmdFind_Click()
```

```
Dim searchTerm As String
searchTerm = TextBoxSearch.Text

Dim foundIndex As Long
foundIndex = -1

' Loop through ListBox items to find a match
For i = 0 To DataListBox.ListCount - 1
    For j = 0 To DataListBox.ColumnCount - 1
        If DataListBox.List(i, j) = searchTerm Then
            foundIndex = i
            Exit For
        End If
    Next j

    If foundIndex >= 0 Then
        Exit For
    End If
Next i

If foundIndex >= 0 Then
    DataListBox.Selected(foundIndex) = True
    DataListBox.TopIndex = foundIndex ' Scroll to the found item
    MsgBox searchTerm & " is found at index: " & foundIndex
Else
```

```
    MsgBox searchTerm & " not found."
  End If
```

This code defines a subroutine named CmdFind_Click that is executed when the "Find" button (presumably named CmdFind) is clicked. The subroutine searches for a specific term entered in a TextBox within a ListBox and provides feedback about the search result. Here's a step-by-step explanation of the code:

CmdFind_Click Subroutine:

This subroutine is triggered when the "Find" button is clicked.

Search Term Setup:

The content of the TextBox named TextBoxSearch is assigned to the searchTerm variable. This is the term you want to search for in the ListBox.

Initialization of foundIndex:

The foundIndex variable is initialized to -1. This variable will be used to store the index of the found item in the ListBox.

Looping Through ListBox Items:

Two nested loops iterate through each cell in the ListBox's rows and columns.

The DataListBox.List(i, j) syntax accesses the value in the i-th row and j-th column of the ListBox's data.

Finding a Match:

If a match is found between the current ListBox cell's value and the searchTerm, the foundIndex is set to the index of the matching row.

The Exit For statement terminates the inner loop once a match is found.

Exiting the Loop:

If a match is found (foundIndex >= 0), the outer loop is also exited using another Exit For statement.

Displaying Search Result:

If a match is found, the corresponding item in the ListBox is selected and the ListBox is scrolled to make the item visible.

A message box displays the search term and the index of the found item.

If no match is found, a message box informs the user that the search term was not found.

In summary, this subroutine allows the user to search for a specific term within a ListBox, highlights the matching item if found, and provides feedback about the search result using message boxes.

Below code is the Continue from above code…

```
Dim wsProd As Worksheet
    Set wsProd = ThisWorkbook.Worksheets("Product_Master")

    Dim lastProdRow As Long
    lastProdRow = wsProd.Cells(wsProd.Rows.Count, "A").End(xlUp).row

    Dim rngProd As Range
    Set rngProd = wsProd.Range("A2:B" & lastProdRow)


    Dim wsCust As Worksheet
    Set wsCust = ThisWorkbook.Worksheets("Customer_Master")

    Dim lastCustRow As Long
    lastCustRow = wsCust.Cells(wsCust.Rows.Count, "A").End(xlUp).row
```

```
    Dim rngCust As Range
    Set rngCust = wsCust.Range("A2:B" & lastCustRow)

    Dim wsSR As Worksheet
    Set wsSR = ThisWorkbook.Worksheets("SR_Master")

    Dim lastSRRow As Long
    lastSRRow = wsSR.Cells(wsSR.Rows.Count, "A").End(xlUp).row

    Dim rngsr As Range
    Set rngsr = wsSR.Range("A2:B" & lastSRRow)

    LblCustID.Caption = DataListBox.List(DataListBox.ListIndex, 0)
    LblInvNo.Caption = DataListBox.List(DataListBox.ListIndex, 1)
    LblInvDate.Caption = DataListBox.List(DataListBox.ListIndex, 2)
    LblInvAmt.Caption = "Rs. " & Format(DataListBox.List(DataListBox.ListIndex, 3), "###,##0.00")
    LblProdID.Caption = DataListBox.List(DataListBox.ListIndex, 5)
    LBLSRID.Caption = DataListBox.List(DataListBox.ListIndex, 8)

    Dim LblPName As String, TxtCustName As String, TxtSRName As String

    For k = 1 To lastCustRow
        If wsCust.Cells(k, 1).value = LblCustID.Caption Then
            'If wsCust.Cells(i, 8).value = "Y" Then
                'TotalProdY = TotalProdY + wsCust.Cells(i, "D").value
```

```
        'Else
            'TotalProdN = TotalProdN + wsCust.Cells(i, "D").value
        'End If
        TxtCustName = wsCust.Cells(k, "B").value
     End If
    'TotalProdYN = TotalProdYN + TotalProdY
    Next k
LblCName.Caption = "- " & TxtCustName


 For i = 1 To lastProdRow
    If wsProd.Cells(i, 1).value = LblProdID.Caption Then
        'If wsCust.Cells(i, 8).value = "Y" Then
            'TotalProdY = TotalProdY + wsCust.Cells(i, "D").value
        'Else
            'TotalProdN = TotalProdN + wsCust.Cells(i, "D").value
        'End If
        LblPName = wsProd.Cells(i, "B").value
     End If
    'TotalProdYN = TotalProdYN + TotalProdY
    Next i
LblProdName.Caption = "- " & LblPName

 For j = 1 To lastSRRow
    If wsSR.Cells(j, 1).value = LBLSRID.Caption Then
```

```
        'If wsCust.Cells(i, 8).value = "Y" Then
            'TotalProdY = TotalProdY + wsCust.Cells(i, "D").value
        'Else
            'TotalProdN = TotalProdN + wsCust.Cells(i, "D").value
        'End If
        TxtSRName = wsSR.Cells(j, "B").value
      End If
     'TotalProdYN = TotalProdYN + TotalProdY
     Next j
  LblSRName.Caption = "- " & TxtSRName

End Sub
```

This code appears to be part of a subroutine that updates various labels on a user form based on the selected item in a ListBox. Here's a breakdown of the code:

Worksheets Setup:

Separate worksheets are defined for "Product_Master," "Customer_Master," and "SR_Master."

Last rows for each worksheet are calculated to determine the range of data.

Setting Ranges:

Ranges (rngProd, rngCust, and rngsr) are defined to represent the relevant columns for each respective worksheet.

Setting Label Captions:

The code proceeds to update various labels on the user form based on the selected item in the ListBox (DataListBox).

Updating Labels for Customer Information:

The LblCustID.Caption is updated with the value from the first column (Column A) of the selected item.

The LblCName.Caption is updated with the corresponding customer name fetched from the "Customer_Master" worksheet using a loop.

Updating Labels for Product Information:

The LblProdID.Caption is updated with the value from the sixth column (Column F) of the selected item.

The LblProdName.Caption is updated with the corresponding product name fetched from the "Product_Master" worksheet using a loop.

Updating Labels for Sales Representative Information:

The LBLSRID.Caption is updated with the value from the ninth column (Column I) of the selected item.

The LblSRName.Caption is updated with the corresponding sales representative name fetched from the "SR_Master" worksheet using a loop.

Label Formatting:

Each label's caption is prefixed with a dash (" - ") for visual separation.

Looping Through Worksheet Data:

The loops through the customer, product, and sales representative data use variables (k, i, and j) to iterate through rows.

If a match is found between the identifier in the ListBox and the data in the respective worksheet, the corresponding name is fetched.

The purpose of this code is to display additional details related to the selected item in the ListBox using information stored in different worksheets. The customer name, product name, and sales representative name are fetched based on the identifiers associated with the selected item. The fetched names are then displayed with appropriate labels on the user form.

Listbox Name is DataListBox

```
Private Sub DataListDisplay()

    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Data")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    Dim rng As Range
    Set rng = ws.Range("A1:L" & lastRow) ' Assuming data starts from A2

    DataListBox.Clear
    DataListBox.List = rng.value

    TotRecord.Caption = "Total Records: " & lastRow - 1

    Dim TotalY, TotalN, TotalYN
    Dim i As Integer
    Dim YesNo As String
    YesNo = "Y"

    For i = 2 To lastRow
       If ws.Cells(i, 8).value = YesNo Then
        TotalY = TotalY + ws.Cells(i, "D").value
       Else
```

```
        TotalN = TotalN + ws.Cells(i, "D").value
         End If
        TotalYN = TotalYN + ws.Cells(i, "D").value
        Next i

    TotAmt.Caption = "Rs. " & Format(TotalYN, "###,##0.00")

End Sub
```

This code populates a list box (DataListBox) with data from a specific worksheet and range. It also calculates and displays the sum of "Amount" values based on a condition in the "Data" worksheet. Here's an explanation of the code:

Worksheet Setup:

The code sets up the "Data" worksheet to work with using the ws variable.

Last Row Calculation:

The last used row in column A of the "Data" worksheet is determined (lastRow).

Setting Range and Populating List Box:

The range of data (rng) is defined to cover columns A to L, from row 1 to the last row.

The code clears any existing items in the DataListBox and then populates it with data from the defined range.

Displaying Total Records:

The TotRecord label caption is updated with the count of total records in the "Data" worksheet.

Calculating and Displaying Totals:

The code initializes variables to keep track of totals: TotalY, TotalN, and TotalYN.

A loop runs from row 2 to the last row of the "Data" worksheet.

If the value in column H ("Payment_Status") is "Y," the corresponding "Amount" value (column D) is added to TotalY. Otherwise, it's added to TotalN.

The "Amount" value is always added to TotalYN.

After the loop, the TotAmt label caption is updated to display the total of all "Amount" values (TotalYN) formatted as currency.

This code effectively populates the list box with data and calculates and displays the total "Amount" values based on the specified conditions.

```
Private Sub Label3_Click()
    Unload Me
End Sub

Private Sub UserForm_Initialize()
    DataListDisplay
End Sub
```

These event procedures are part of a UserForm in Excel VBA. Let's break down what each procedure does:

Label3_Click:

This event procedure is triggered when the user clicks on Label3 in the UserForm.

Label3 is typically used as a close or exit button.

The code uses the Unload Me statement to close the UserForm.

UserForm_Initialize:

This event procedure is triggered when the UserForm is initialized, i.e., when it is shown to the user.

The DataListDisplay subroutine is called within this event procedure.

DataListDisplay is presumably a subroutine that populates the UserForm's list box with data.

In summary, when the UserForm is initialized, it automatically populates the list box with data using the DataListDisplay subroutine. If the user clicks on Label3, the UserForm is closed. This provides a simple way to display data and close the UserForm when needed.

Gautam Banerjee

“Helping beginners learn something new is a great way to share your knowledge and make a positive impact”.

Email: gincom1@yahoo.com

If you have any queries, please visit our "Contact Us" page.

Thank You. See you again!

PLEASE DONATE

Gautam Banerjee

Age: 63

Pay by UPI

9748327614