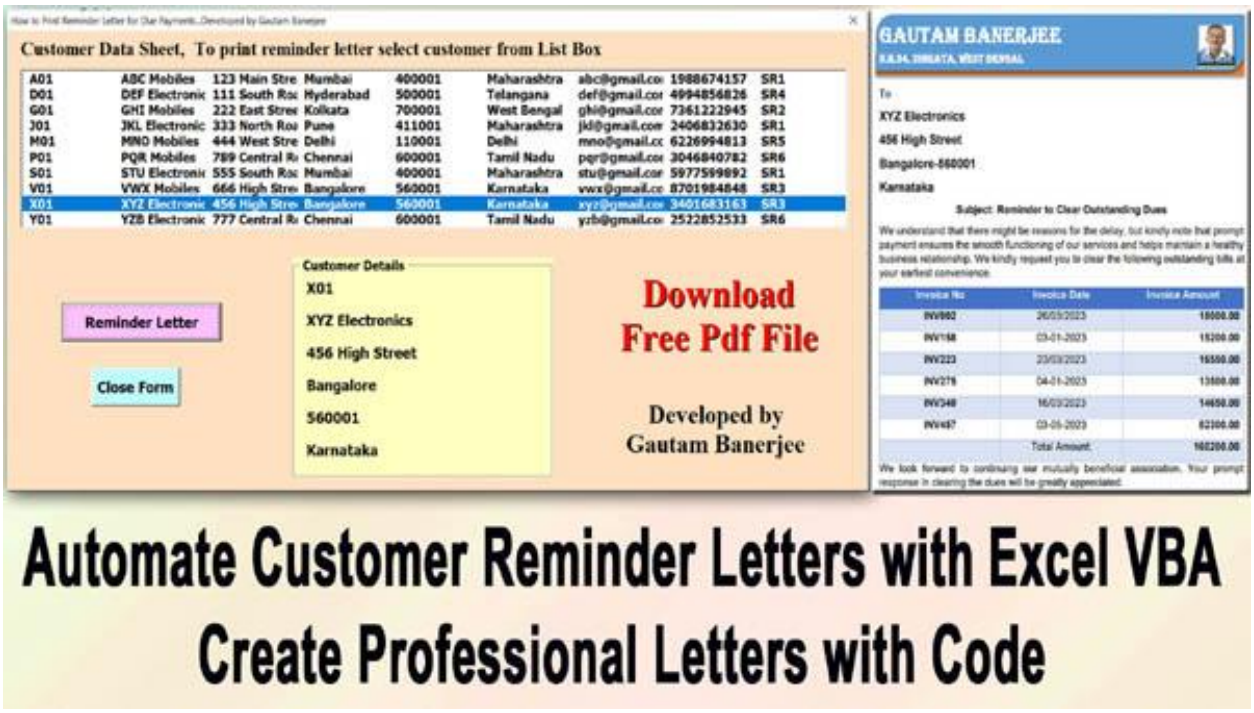

: Are you tired of manually sending reminder letters to customers for outstanding dues? In this tutorial, you'll learn how to automate the process using Excel VBA. Say goodbye to time-consuming manual tasks and hello to efficient, professional, and personalized reminder letters.

In this step-by-step guide, we'll walk you through the process of creating a dynamic reminder letter template that fetches customer data, inserts it into a Word document, and even calculates and displays the total outstanding amount. Whether you're a business owner, freelancer, or simply looking to enhance your Excel skills, this tutorial is for you.

🔥 What You'll Learn:

Set up the template: Create a dynamic Word document template with placeholders.

Fetch customer data: Use Excel VBA to retrieve customer information from your dataset.

Personalize letters: Automatically fill in customer details like name, address, and more.

Calculate totals: Learn how to calculate and display the total outstanding amount.

Automation benefits: Save time, improve accuracy, and send professional letters effortlessly.

By the end of this tutorial, you'll have a powerful tool in your hands to streamline your customer communication process and enhance your Excel VBA skills. Don't miss out on this opportunity to simplify your workflow and make a positive impact on your business. Let's dive in and revolutionize the way you send reminder letters.

🚀 Download the code and resources:

www.gincom1/wixsite\index.com

Stay tuned for more tutorials and tips to supercharge your Excel productivity! If you found this video helpful, don't forget to like, share, and subscribe for more content.

Insert One List Box, Two Command Buttons and Few Labels. Change name and caption of List box and command buttons from Property.

Create a Word Document and save it in the same folder as your Excel Sheets, or you can opt to create a sub-folder specifically for Reminder letters.

Start Coding :

```
Dim LblCustID As String, CustName As String, CustAdd As String
Dim CustCity As String, CustPIN As String, CustState As String
Dim TolAmt As Long
```

Declaring variables at the top of the code window is a common practice in programming and has several advantages:

Readability: By declaring variables at the beginning of your code, it's easier for you and other developers to understand the data that will be used in the program.

Consistency: It creates a consistent structure in your code, making it easier to maintain and modify.

Prevention of Errors: Declaring variables upfront helps in avoiding issues like using a variable before it's assigned a value or mistakenly using the same variable name for different purposes.

Documentation: It serves as a form of documentation, providing a clear list of the variables being used in your program.

In this case, the code declares several variables at the beginning. This approach is recommended as it helps you keep track of the variables you'll be using throughout the program. It's a good practice that can make your code more organized and less error-prone.

Command Button name is : CmdExt

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

The "CmdExt_Click" subroutine is designed to close or unload the current user form when the corresponding button, likely named "CmdExt," is clicked. This action helps provide a seamless user experience by allowing users to exit or dismiss the form as needed.

Insert List Box and Name it : CustList

```
Private Sub CustListDisplay()

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

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

    Dim rng As Range
    Set rng = wsCust.Range("A2:J" & lastRow) ' Assuming data starts from A2

    CustList.Clear
    CustList.List = rng.value

End Sub
```

The "CustListDisplay" subroutine is designed to populate a list box control named "CustList" with customer data retrieved from the "Customer_Master" worksheet. This subroutine helps display customer information in an organized manner, allowing users to conveniently view and interact with the data within the list box.

Here's how the subroutine works:

It first identifies the worksheet named "Customer_Master" within the current workbook using the "Set wsCust = ThisWorkbook.Worksheets("Customer_Master")" line.

It then determines the last populated row in column A of the "Customer_Master" worksheet using "lastRow =

wsCust.Cells(wsCust.Rows.Count, "A").End(xlUp).Row". This helps determine the range of data to be extracted.

The "rng" range is set to cover the data range from cell A2 to column J and the last populated row using "Set rng = wsCust.Range("A2:J" & lastRow)".

The "CustList" list box control is cleared using "CustList.Clear" to ensure a fresh start for populating data.

The data from the "rng" range is then assigned to the "CustList" list box control using "CustList.List = rng.Value". This populates the list box with the customer data retrieved from the specified range.

By executing the "CustListDisplay" subroutine, you're facilitating the display of customer information within the "CustList" list box, providing users with an organized and accessible way to view the data.

```
Private Sub UserForm_Initialize()
    CustListDisplay
    CmdReminder.Enabled = False
End Sub
```

In the "UserForm_Initialize" event handler, you're initializing the user form and setting up its initial state. Let's break down what each part of this code does:

CustListDisplay: This line calls the "CustListDisplay" subroutine, which populates the "CustList" list box with customer data from the "Customer_Master" worksheet. This ensures that when the user form is initialized, the list box is populated with the relevant customer information.

CmdReminder.Enabled = False: This line disables (sets to "False") the "CmdReminder" command button. This button is presumably used for sending reminders, but since you have disabled it in the initialization, it won't be clickable or usable until you enable it later in your code.

Overall, this initialization routine sets up the user form by loading customer data into the list box and disabling the reminder command button at the start. This ensures that users have access to the customer information but can't trigger the reminder functionality until certain conditions are met in your application's workflow.

```
Private Sub CustList_Click()
    CustID = CustList.List(CustList.ListIndex, 0)
    CustName = CustList.List(CustList.ListIndex, 1)
    CustAdd = CustList.List(CustList.ListIndex, 2)
    CustCity = CustList.List(CustList.ListIndex, 3)
    CustPIN = CustList.List(CustList.ListIndex, 4)
    CustState = CustList.List(CustList.ListIndex, 5)
    Label1.Caption = CustID
    Label2.Caption = CustName
    Label3.Caption = CustAdd
    Label4.Caption = CustCity
    Label5.Caption = CustPIN
    Label6.Caption = CustState
    CmdReminder.Enabled = True
End Sub
```

This CustList_Click event handler is triggered when an item in the CustList list box is clicked by the user. It performs the following actions:

It retrieves the customer details (ID, name, address, city, PIN, and state) from the selected item in the list box using the List property and assigns them to the respective variables (CustID, CustName, CustAdd, CustCity, CustPIN, CustState).

It updates the caption of six labels (Label1 through Label6) with the corresponding customer details.

It enables the CmdReminder command button, allowing the user to trigger the reminder functionality for the selected customer.

This event handler essentially provides a dynamic way of displaying customer details when the user clicks on a customer's name in the list box, and it prepares the user interface for sending a reminder to the selected customer.

Insert a Command Button and Name it : CmdReminder

```
Private Sub CmdReminder_Click()

    Dim WApp As New Word.Application
    Dim WDoc As Word.Document
    WApp.Visible = True
    Set WDoc = WApp.Documents.Open(ThisWorkbook.Path & "\Reminder" & ".docx")

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

This CmdReminder_Click event handler is triggered when the user clicks the CmdReminder button. It initiates the process of creating reminder letters for selected customers. Let's break down the code:

It creates a new instance of the Word application (WApp) and sets it to be visible.

It opens the Word document named "Reminder.docx" located in the same directory as the Excel workbook. This document will serve as the template for the reminder letters.

It declares a worksheet variable (wsData) and sets it to refer to the "Data" worksheet in the workbook. This worksheet is assumed to contain the data related to customers and their payment statuses.

The code is setting up the environment for generating reminder letters by opening the Word document template and establishing a connection to the Excel data. The following steps will involve replacing placeholders in the Word document with actual customer data and performing other necessary operations to complete the reminder letters.

```
With WDoc.Content.Find
        .Text = "#CustName#"
        .Replacement.Text = CustName
        .Execute Replace:=wdReplaceAll
        .Text = "#CustomerAddress#"
        .Replacement.Text = CustAdd
        .Execute Replace:=wdReplaceAll
        .Text = "#City#"
        .Replacement.Text = CustCity
        .Execute Replace:=wdReplaceAll
        .Text = "#PIN#"
        .Replacement.Text = CustPIN
        .Execute Replace:=wdReplaceAll
        .Text = "#State#"
        .Replacement.Text = CustState
        .Execute Replace:=wdReplaceAll
```

```
        .Text = "<TotalAmt>"
        .Replacement.Text = "TotAmt#"
        .Execute Replace:=wdReplaceAll

    End With
```

This part of the code is responsible for replacing placeholders in the Word document template with actual customer data and the total amount. Let's break down what each line does:

.Text = "#CustName#": Sets the search text to the placeholder "#CustName#".

.Replacement.Text = CustName: Sets the replacement text to the value stored in the CustName variable.

.Execute Replace:=wdReplaceAll: Executes the search and replace operation for the currently set text.

This process is repeated for each placeholder in the document: "#CustomerAddress#", "#City#", "#PIN#", "#State#", and "<TotalAmt>".

However, there's a small error in the replacement text for "<TotalAmt>": .Replacement.Text = "TotAmt#" should be .Replacement.Text = TotAmt.

This way, the placeholder "<TotalAmt>" will be replaced with the value stored in the TotAmt variable.

Once these replacements are done, the Word document will have placeholders replaced with actual customer data.

The next step would involve performing the operations needed to calculate the total amount (TotAmt) and then replacing the "<TotalAmt>" placeholder with this calculated total amount.

'This is comment not code

Dim TableRow As Long ***' Keep track of the current row index in the table***

TableRow = 2 ' Start populating data from the second row

TableRow: This variable is used to keep track of the current row index in the Word table where data is being populated. It starts from the second row (index 2) because the first row is typically reserved for headers. As data is added to the table, the TableRow value is incremented to ensure that each new piece of data is placed in the correct row.

TotalAmount: This variable is used to calculate the total amount of invoices for the selected customer with payment status "N". It accumulates the invoice amounts as the loop iterates through the data.

These definitions help manage the placement of data in the Word table and calculate the total amount for the reminder letter.

```
Dim TotalAmount As Double ' To calculate the total amount


' Define the column numbers for relevant information (adjust as needed)
Dim CustIDCol As Long
Dim PaymentStatusCol As Long
Dim InvNoCol As Long
Dim InvDt As Long
Dim InvAmt As Long
```

Create variables to store field value.

```
CustIDCol = 1 ' For example, column A
    PaymentStatusCol = 8 ' For example, column H
    InvNoCol = 2 ' For example, column C
    InvDt = 3
    InvAmt = 4
```

Store data of Column Number from Excel sheet.

```
' Customer ID for which you want to print the letter
```

```
    Dim TargetCustomerID As String
    TargetCustomerID = Label1.Caption
```

TargetCustomerID: This variable holds the Customer ID for which you want to print the reminder letter. It takes its value from the caption of Label1, which is set when a customer is selected from the CustList ListBox.

```
    ' Filter and process data
    Dim LastDataRow As Long
    LastDataRow = wsData.Cells(wsData.Rows.Count, CustIDCol).End(xlUp).row
```

LastDataRow: This variable is used to determine the last row containing data in the wsData worksheet. It finds the last row by searching from the bottom of the worksheet using the End(xlUp) method starting from the first column (CustIDCol). This helps ensure that you're processing all the relevant rows of data.

```
    Dim tbl As Word.table
    Set tbl = WDoc.Tables(1)
```

tbl: This variable holds a reference to the first table in the Word document (WDoc). This table is where you'll be populating the data from the wsData worksheet.

These declarations and definitions are crucial for filtering and processing the data for the selected customer ID and payment status, and for working with the Word table where the reminder letter content will be added.

For Datarow = 2 To LastDataRow **' Assuming data starts from row 2**

For Datarow = 2 To LastDataRow: This is a loop that will iterate from row 2 to the value of LastDataRow. It means that the loop will go through all rows of data in the wsData worksheet, starting from row 2 (assuming that's where your data starts) and going up to the last row containing data (LastDataRow).

Datarow is the loop variable that will take on values from 2 to LastDataRow. It represents the current row being processed in the loop.

The purpose of this loop is to go through each row of data in the worksheet and perform the necessary operations, such as filtering based on the customer ID and payment status, and adding relevant data to the Word document's table.

```
Dim CustID As String
    Dim PaymentStatus As String
    Dim CustAddress As String
    Dim InvNoCol2 As String
    Dim InvDtCol3 As String
    Dim InvAmtCol4 As String

    CustID = wsData.Cells(Datarow, CustIDCol).value
    PaymentStatus = wsData.Cells(Datarow, PaymentStatusCol).value
    InvNoCol2 = wsData.Cells(Datarow, InvNoCol).value
    InvDtCol3 = wsData.Cells(Datarow, InvDt).value
    InvAmtCol4 = wsData.Cells(Datarow, InvAmt).value
```

The code declares and initializes several variables to store different pieces of information for each row of data being processed in the loop. Here's what each variable represents:

CustID: Stores the value of the customer ID for the current row.

PaymentStatus: Stores the value of the payment status for the current row.

CustAddress: This variable seems to be declared but not used in the provided snippet.

InvNoCol2: Stores the value of the invoice number for the current row.

InvDtCol3: Stores the value of the invoice date for the current row.

InvAmtCol4: Stores the value of the invoice amount for the current row.

Each of these variables is assigned a value from the corresponding cell in the wsData worksheet, based on the row index (Datarow) and the column index (e.g., CustIDCol, PaymentStatusCol, etc.). These values are used later in the code to populate the Word document's table and perform filtering based on the customer ID and payment status.

```
If CustID = TargetCustomerID And PaymentStatus = "N" Then
        tbl.Rows.Add ' Add a new row to the table

        ' Populate the cells with data
        tbl.Cell(TableRow, 1).Range.Text = InvNoCol2
        tbl.Cell(TableRow, 2).Range.Text = InvDtCol3
        tbl.Cell(TableRow, 3).Range.Text = Format(InvAmtCol4, "0.00")

        TotalAmount = TotalAmount + InvAmtCol4 ' Calculate the total amount
        TableRow = TableRow + 1
    End If
```

```
    Next Datarow
```

This part of the code is responsible for populating the Word document's table with the filtered data for the specific customer ID and payment status "N." Here's what each section does:

The If statement checks whether the current row's CustID matches the TargetCustomerID and if the PaymentStatus is "N." This is the filtering condition to determine whether the current row's data should be added to the Word document's table.

If the condition is met, a new row is added to the Word document's table using tbl.Rows.Add.

The data from the relevant variables (InvNoCol2, InvDtCol3, and InvAmtCol4) is then populated into the corresponding cells of the table using the tbl.Cell method. The Format function is used to format the invoice amount to have two decimal places.

The TotalAmount variable is updated by adding the current row's invoice amount (InvAmtCol4) to it. This variable keeps track of the total amount for the customer.

TableRow is incremented to prepare for populating the next row in the table.

Overall, this section of the code iterates through the filtered data, adds the relevant rows to the Word document's table, populates the cells with the data, and calculates the total amount for the customer.

```
    ' Add the total amount to the last row of the table
    tbl.Cell(TableRow, 2).Range.Text = "Total Amount: "
    tbl.Cell(TableRow, 3).Range.Text = Format(TotalAmount, "0.00")
    TotAmt = Format(TotalAmount, "0.00")
```

```
    WDoc.ExportAsFixedFormat ThisWorkbook.Path & "\" &
TargetCustomerID & ".pdf", wdExportFormatPDF
    WDoc.SaveAs2 ThisWorkbook.Path & "\" & TargetCustomerID & ".docx"
    WDoc.Close

    WApp.Quit
    Set WDoc = Nothing
    Set WApp = Nothing
```

he tbl.Cell(TableRow, 2).Range.Text and tbl.Cell(TableRow, 3).Range.Text lines add the "Total Amount" label and the formatted total amount to the last row of the table.

The TotAmt = Format(TotalAmount, "0.00") line assigns the formatted total amount to the TotAmt variable.

WDoc.ExportAsFixedFormat exports the Word document as a PDF file using the customer's ID as the file name.

WDoc.SaveAs2 saves the Word document with the customer's ID as the file name.

WDoc.Close closes the Word document.

WApp.Quit closes the Word application.

The Set statements are used to clear the object variables (WDoc and WApp) to release resources.

This code completes the process of generating the reminder letter, populating the data and total amount, saving the documents, and then closing the Word application.

```
MsgBox "Reminder letter generated successfully!", vbInformation,
"Success...Gautam Banerjee"
```

This line displays a message box to the user indicating that the reminder letter was generated successfully.

```
On Error GoTo ErrorHandler ' Enable error handling

On Error GoTo 0
    Exit Sub  ' Exit the sub here to prevent error handling from executing

ErrorHandler:
    ' Handle errors here
    MsgBox "An error occurred: " & Err.Description, vbExclamation, "Error"
    ' Optionally, log the error or take appropriate action
    Resume Next ' Continue execution after handling the error
```

On Error GoTo ErrorHandler: This line enables error handling and tells VBA to jump to the ErrorHandler label when an error occurs.

On Error GoTo 0: This line disables error handling. After successful execution, you want to turn off error handling.

Exit Sub: This line exits the CmdReminder_Click procedure to prevent error handling from executing when there's no error.

ErrorHandler:: This is a label indicating the start of the error handling section.

MsgBox "An error occurred: " & Err.Description, vbExclamation, "Error": This line displays an error message to the user, including the description of the error.

Resume Next: This line tells VBA to continue execution after handling the error.

You can customize the error handling code to match your specific needs, such as logging errors or providing more detailed error messages.

Remember that error handling should be tailored to the context of your application and the potential errors that might occur.

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

Copy and Paste Below Full Coding :

```
Dim LblCustID As String, CustName As String, CustAdd As String
Dim CustCity As String, CustPIN As String, CustState As String
Dim TolAmt As Long

Private Sub CmdExt_Click()
    Unload Me
End Sub

Private Sub CmdReminder_Click()

On Error GoTo ErrorHandler ' Enable error handling

    Dim WApp As New Word.Application
    Dim WDoc As Word.Document
    WApp.Visible = True
    Set WDoc = WApp.Documents.Open(ThisWorkbook.Path & "\Reminder" & ".docx")

    ' Original Data Sheet (Adjust the sheet name as needed)
    Dim wsData As Worksheet
    Set wsData = ThisWorkbook.Worksheets("Data")

        With WDoc.Content.Find
            .Text = "#CustName#"
```

```
        .Replacement.Text = CustName
        .Execute Replace:=wdReplaceAll
        .Text = "#CustomerAddress#"
        .Replacement.Text = CustAdd
        .Execute Replace:=wdReplaceAll
        .Text = "#City#"
        .Replacement.Text = CustCity
        .Execute Replace:=wdReplaceAll
        .Text = "#PIN#"
        .Replacement.Text = CustPIN
        .Execute Replace:=wdReplaceAll
        .Text = "#State#"
        .Replacement.Text = CustState
        .Execute Replace:=wdReplaceAll
        .Text = "<TotalAmt>"
        .Replacement.Text = "TotAmt#"
        .Execute Replace:=wdReplaceAll

    End With

Dim TableRow As Long ' Keep track of the current row index in the table
TableRow = 2 ' Start populating data from the second row

Dim TotalAmount As Double ' To calculate the total amount
```

```
    ' Define the column numbers for relevant information (adjust as needed)
    Dim CustIDCol As Long
    Dim PaymentStatusCol As Long
    Dim InvNoCol As Long
    Dim InvDt As Long
    Dim InvAmt As Long
    ' ... (add more column numbers as needed)

    CustIDCol = 1 ' For example, column A
    PaymentStatusCol = 8 ' For example, column H
    InvNoCol = 2 ' For example, column C
    InvDt = 3
    InvAmt = 4

    ' Customer ID for which you want to print the letter
    Dim TargetCustomerID As String
    TargetCustomerID = Label1.Caption

    ' Filter and process data
    Dim LastDataRow As Long
    LastDataRow = wsData.Cells(wsData.Rows.Count,
CustIDCol).End(xlUp).row

    Dim tbl As Word.table
    Set tbl = WDoc.Tables(1)
```

```
For Datarow = 2 To LastDataRow ' Assuming data starts from row 2
    Dim CustID As String
    Dim PaymentStatus As String
    Dim CustAddress As String
    Dim InvNoCol2 As String
    Dim InvDtCol3 As String
    Dim InvAmtCol4 As String

    CustID = wsData.Cells(Datarow, CustIDCol).value
    PaymentStatus = wsData.Cells(Datarow, PaymentStatusCol).value
    InvNoCol2 = wsData.Cells(Datarow, InvNoCol).value
    InvDtCol3 = wsData.Cells(Datarow, InvDt).value
    InvAmtCol4 = wsData.Cells(Datarow, InvAmt).value


    ' Add a condition to filter based on CustID and PaymentStatus

    If CustID = TargetCustomerID And PaymentStatus = "N" Then
        tbl.Rows.Add ' Add a new row to the table

        ' Populate the cells with data
        tbl.Cell(TableRow, 1).Range.Text = InvNoCol2
        tbl.Cell(TableRow, 2).Range.Text = InvDtCol3
        tbl.Cell(TableRow, 3).Range.Text = Format(InvAmtCol4, "0.00")
        ' ... (populate other columns as needed)
```

```
            TotalAmount = TotalAmount + InvAmtCol4 ' Calculate the total
amount
            TableRow = TableRow + 1
        End If
    Next Datarow

     ' Add the total amount to the last row of the table
    tbl.Cell(TableRow, 2).Range.Text = "Total Amount: "
    tbl.Cell(TableRow, 3).Range.Text = Format(TotalAmount, "0.00")
    TotAmt = Format(TotalAmount, "0.00")


    WDoc.ExportAsFixedFormat ThisWorkbook.Path & "\" &
TargetCustomerID & ".pdf", wdExportFormatPDF
    WDoc.SaveAs2 ThisWorkbook.Path & "\" & TargetCustomerID & ".docx"
    WDoc.Close
    WApp.Quit

    MsgBox "Reminder letter generated successfully!", vbInformation,
"Success...Gautam Banerjee"
    Set WDoc = Nothing
    Set WApp = Nothing

    On Error GoTo 0
    Exit Sub  ' Exit the sub here to prevent error handling from executing
```

```
ErrorHandler:
    ' Handle errors here
    MsgBox "An error occurred: " & Err.Description, vbExclamation, "Error"
    ' Optionally, log the error or take appropriate action
    Resume Next ' Continue execution after handling the error

End Sub

Private Sub CustListDisplay()

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

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

    Dim rng As Range
    Set rng = wsCust.Range("A2:J" & lastRow) ' Assuming data starts from
A2

    CustList.Clear
    CustList.List = rng.value

End Sub
```

```
Private Sub CustList_Click()
    CustID = CustList.List(CustList.ListIndex, 0)
    CustName = CustList.List(CustList.ListIndex, 1)
    CustAdd = CustList.List(CustList.ListIndex, 2)
    CustCity = CustList.List(CustList.ListIndex, 3)
    CustPIN = CustList.List(CustList.ListIndex, 4)
    CustState = CustList.List(CustList.ListIndex, 5)
    Label1.Caption = CustID
    Label2.Caption = CustName
    Label3.Caption = CustAdd
    Label4.Caption = CustCity
    Label5.Caption = CustPIN
    Label6.Caption = CustState
    CmdReminder.Enabled = True
End Sub


Private Sub UserForm_Initialize()
    CustListDisplay
    CmdReminder.Enabled = False
End Sub
```

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