Microsoft Excel Macros & VBA

Report

Excel and VBA Programming

By

Ana Clara Tupinambá Freitas

Oriented by professor Hamid Rajaee

Metro College of Technology, Data Science and Application Program

Contents

Introduction

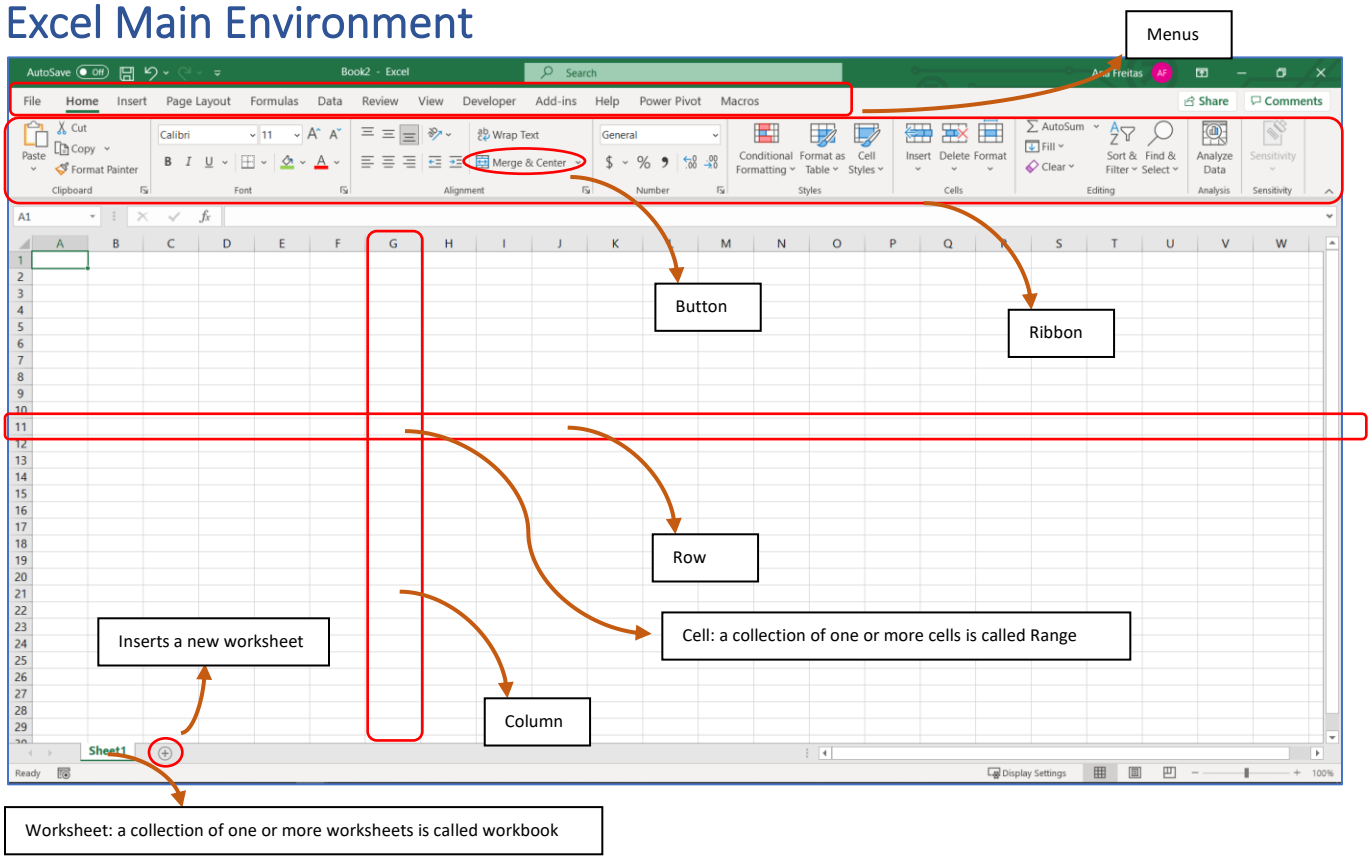
Excel is a major Microsoft Office application for consolidating data and generating reports. VBE furthers Excel's power by providing an environment to automate actions (programming procedures and functions) by using the VBA language program.

Important notes

Menus, buttons, keys, and areas will be encircled by "<>" to emphasize the action to be performed when describing actions in this report.

Although mentioned in conclusions, recording a macro and programming a function are not in the scope of this report.

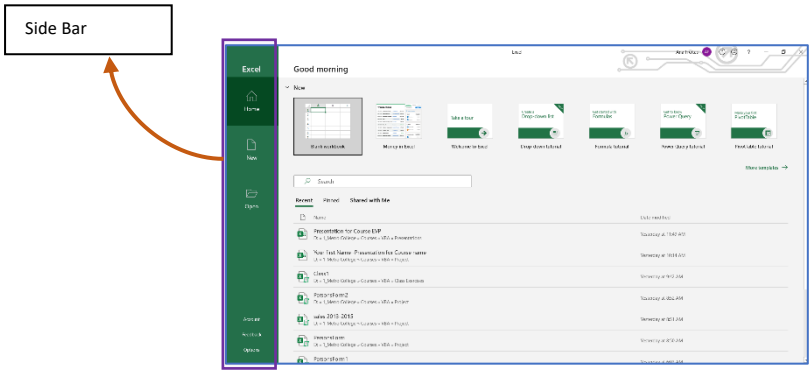
Excel Main Environment



How to open a file

It's similar to other Microsoft Office Applications: With Excel opened, you can Choose:

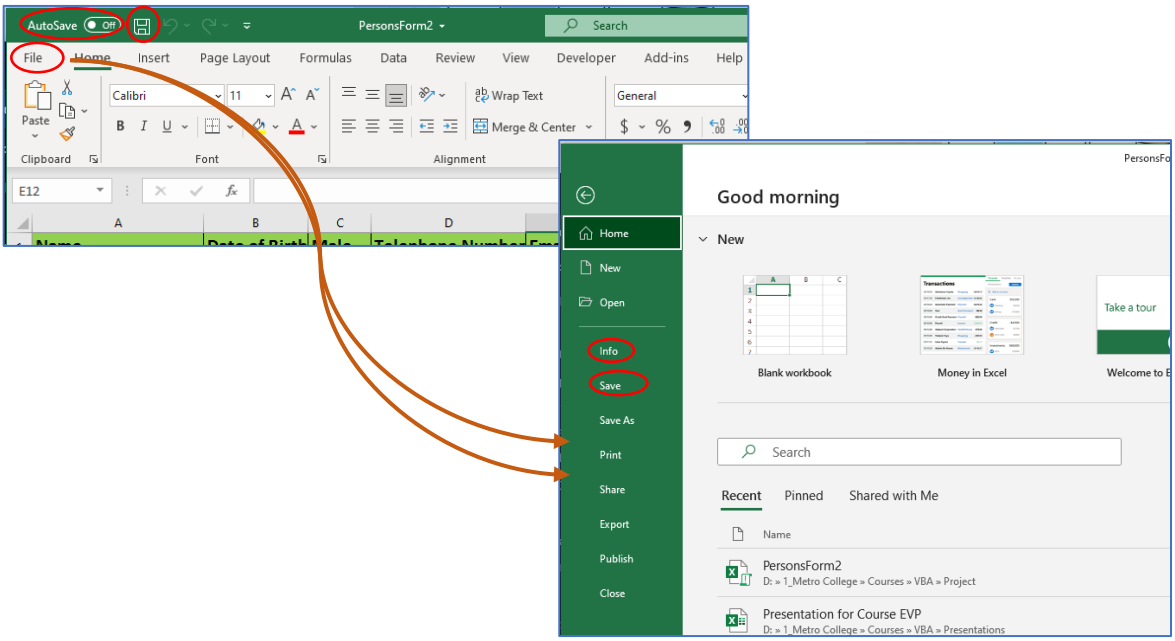
- From the side bar
  - <Home>:
    - From the templates options;
    - From the Recent, Pinned, or Shared with me;
  - <New>:
    - Blank workbook;
    - Search for a template;
    - From the listed templates;
  - <Open>:
    - Recent file;
    - Shared with me;
    - Personal;
    - Other Locations.



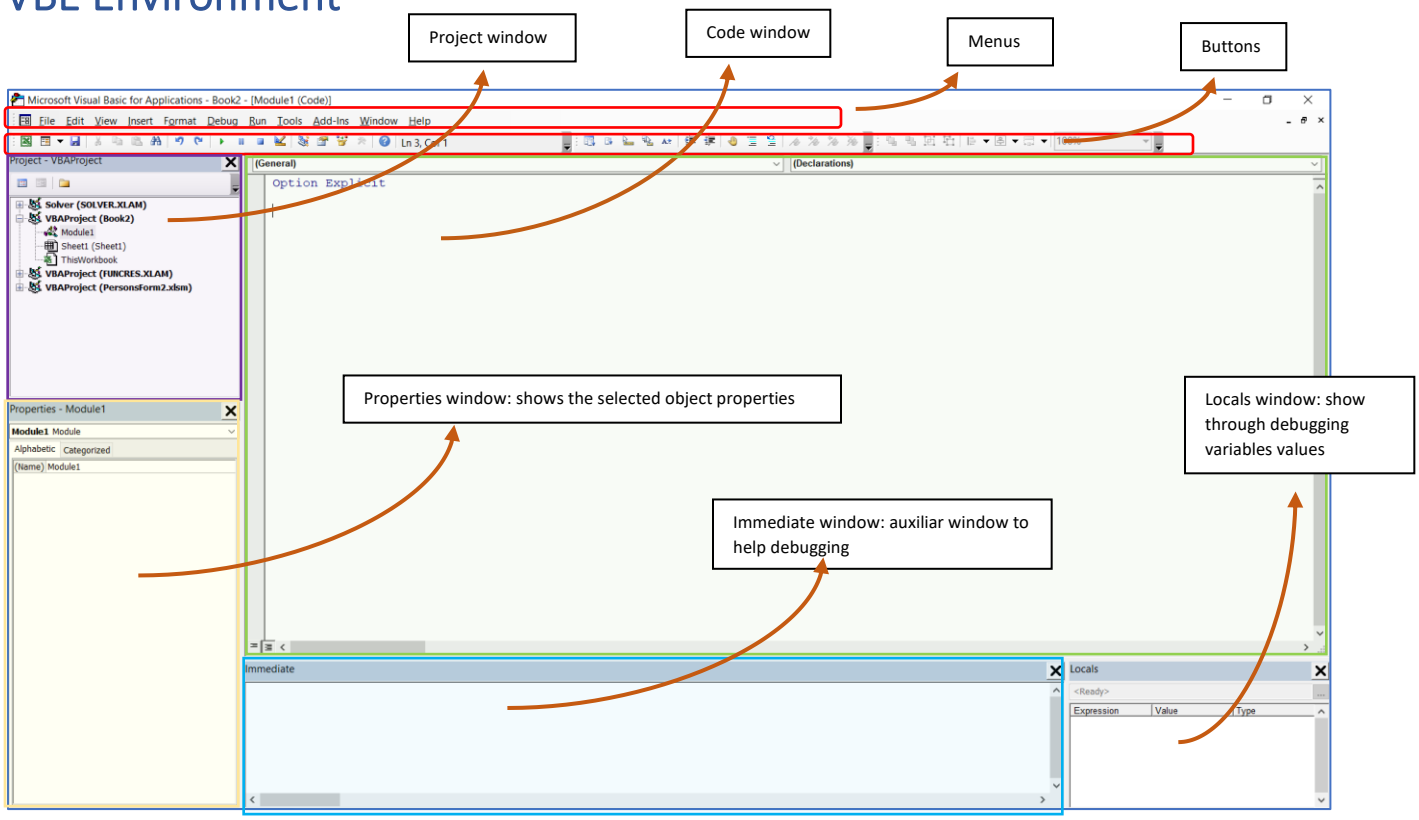
How to save a file

It's similar to other Microsoft Office Applications: With Excel opened, you can Choose:

- AutoSave: saves the file into the Microsoft's onedrive;
- Save button: a shortcut access to the same function encountered in the menu <File>, submenu <Save>;
- Menu <File>, submenu <Save>: You can choose wherever you want to save your file;



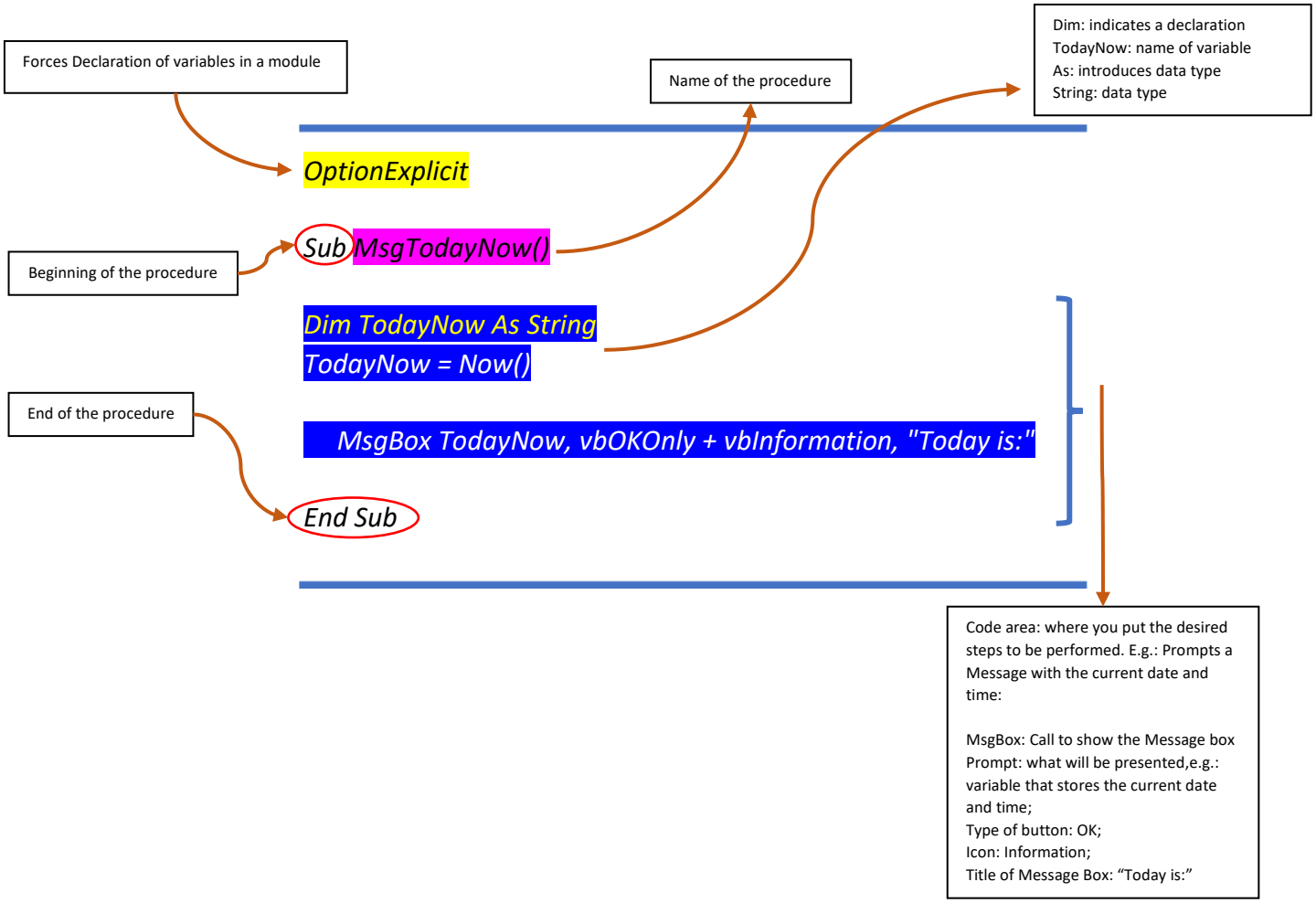
VBE Environment



Anatomy of a procedure

A procedure is used to perform actions and it cannot be used in a worksheet Formula.

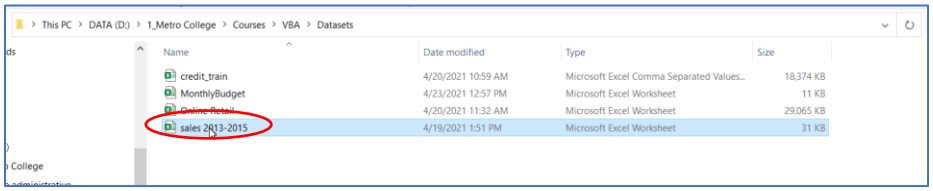
The beginning of procedure id indicated by the reserved word “Sub” followed by its name and parentheses (VBE includes the parentheses automatically). The end is indicated by “End Sub” and your code regarding the procedure comes in-between. E.g.:



Phase 1

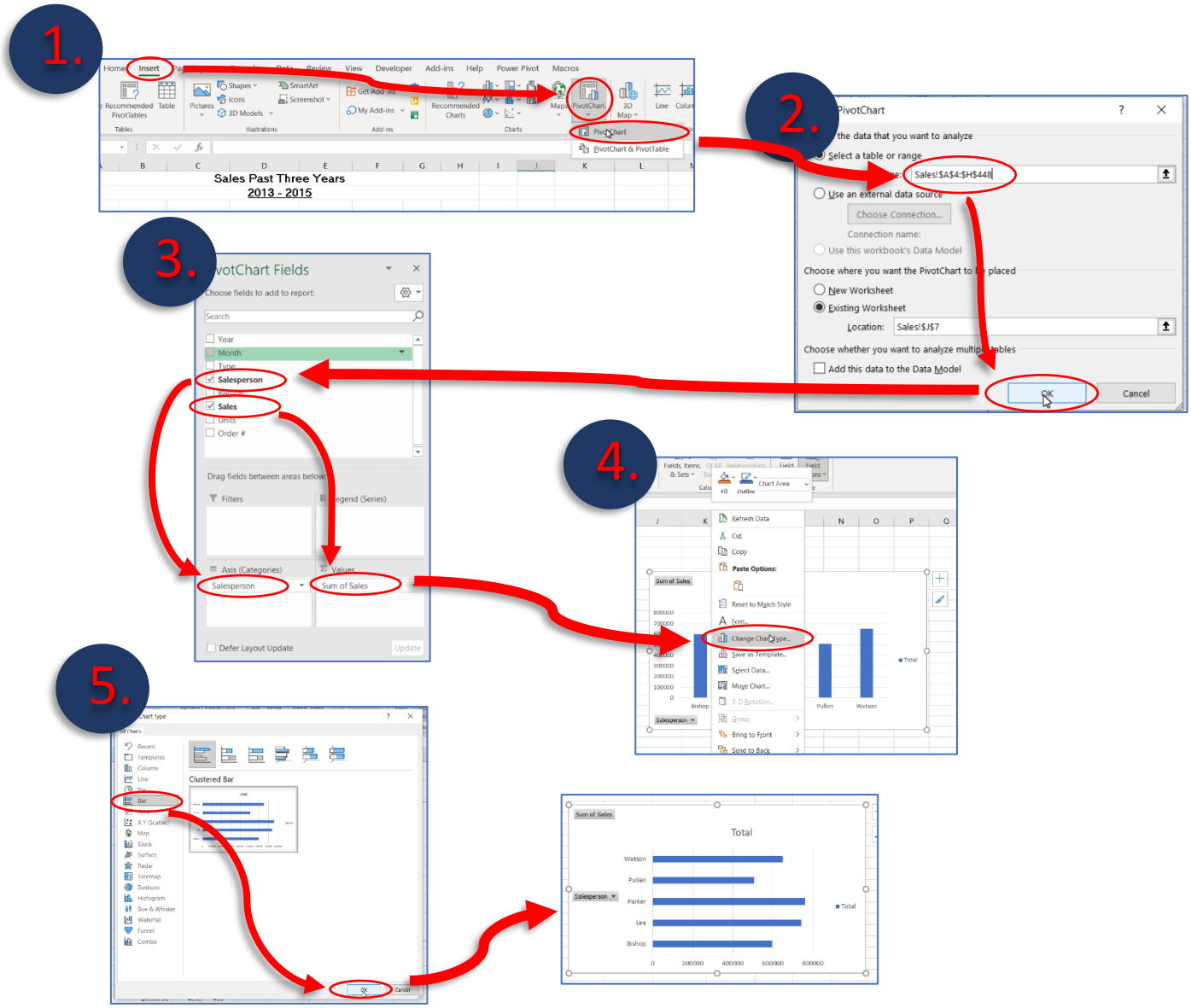
1. Create a bar chart from data of sales in excel file.

After the required dataset (sales 2013-2015.xls) is downloaded, open it by clicking twice on its file.



With the file opened:

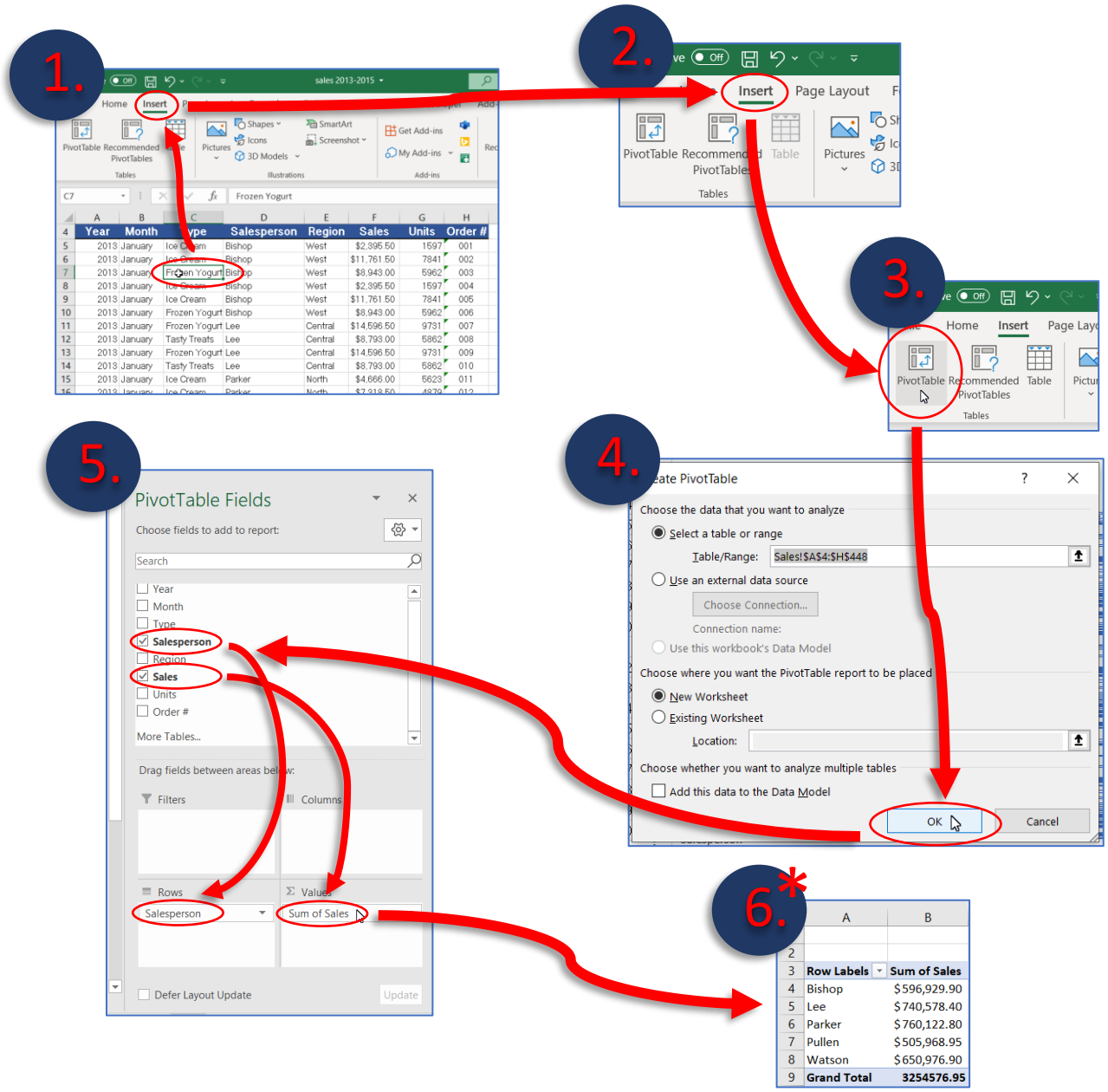
1. Go to the <Insert> menu and Select <PivotChart> on the <Charts> area of the Ribbon;
2. A window to create PivotChart will open. Input the desired range of the table and select where you want to insert the chart: Existing worksheet or a new one;
3. Select your fields in the PivotCharts Fields. A chart will appear;
4. Right-click on the chart an select <Change Chart Type...> ;
5. Select your desired chart type. You chart is ready.



2. Create a pivot table and group data based on salesperson.

1. With the file opened, select any cell within the table;
2. Go to the <Insert> menu;
3. Select PivotTable on the <Tables> area of the Ribbon;
4. A window with information for the creation of the pivot table will open with this information:
   a. Select a table or range: Excel selects the table that contains the previous selected cell is by default.
   b. Choose where you want the Pivot Table report to be placed: New Worksheet.
      *Change this option only if you want to select a location that already exist within the workbook.

   Click <OK>;
5. Select and drag the desired Columns that will form the pivot table on the PivotTable Fields area:
   a. Rows: Salesperson;
   b. Values: Sales
6. The pivot table will be created on the cells area under <Sheet 2>.

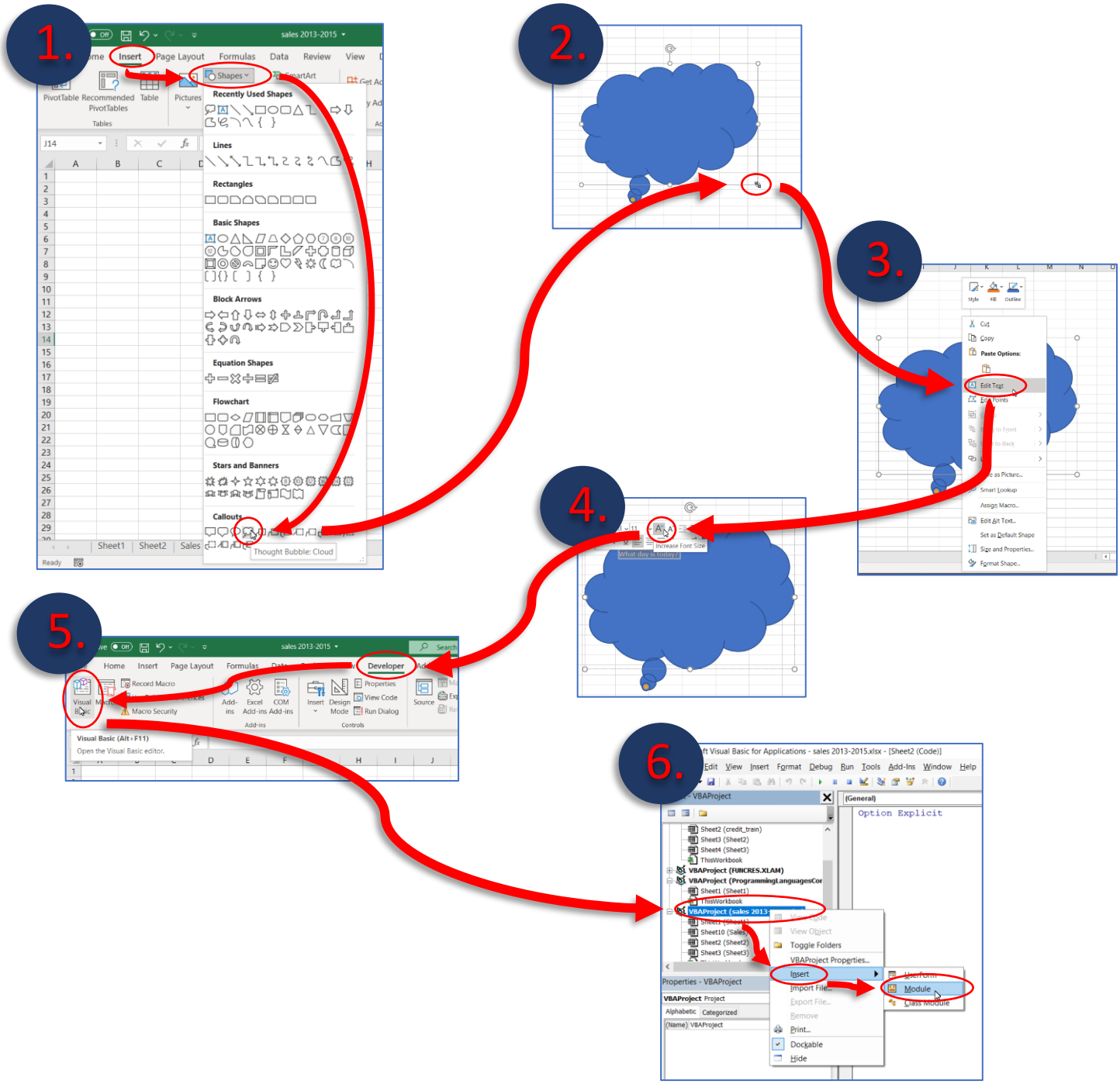


* You can format the values on the pivot table by selecting it and choosing the proper format on the <Number> area of the Ribbon on the <Home> menu.

3. Create a shape object and assign a macro to it that displays in message the current date.

With the worksheet opened, insert a new Sheet as mentioned on the Excel Main Environment, then:

1. Go to the <Insert> menu. Select <Shapes> on the <Illustrations> area of the Ribbon and choose the desired shape;
2. The shape will be inserted on the <Cells> area and it can be resized by clicking on the little circles on the borders (the cursor will change its format to a double pointed arrow), drag it until the desired size is met.
3. You can add a text into the shape by right-clicking it and choosing: <Edit Text> and, then, typing your desired text;
4. After typing the text, you can resize it, by selecting it, and then; clicking the <Increase Font Size> button;
5. Go to the <Developer> menu and click on <Visual Basic> on the <Code> are of the Ribbon. The Visual Basic Editor (VBE) will be opened;
6. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open;

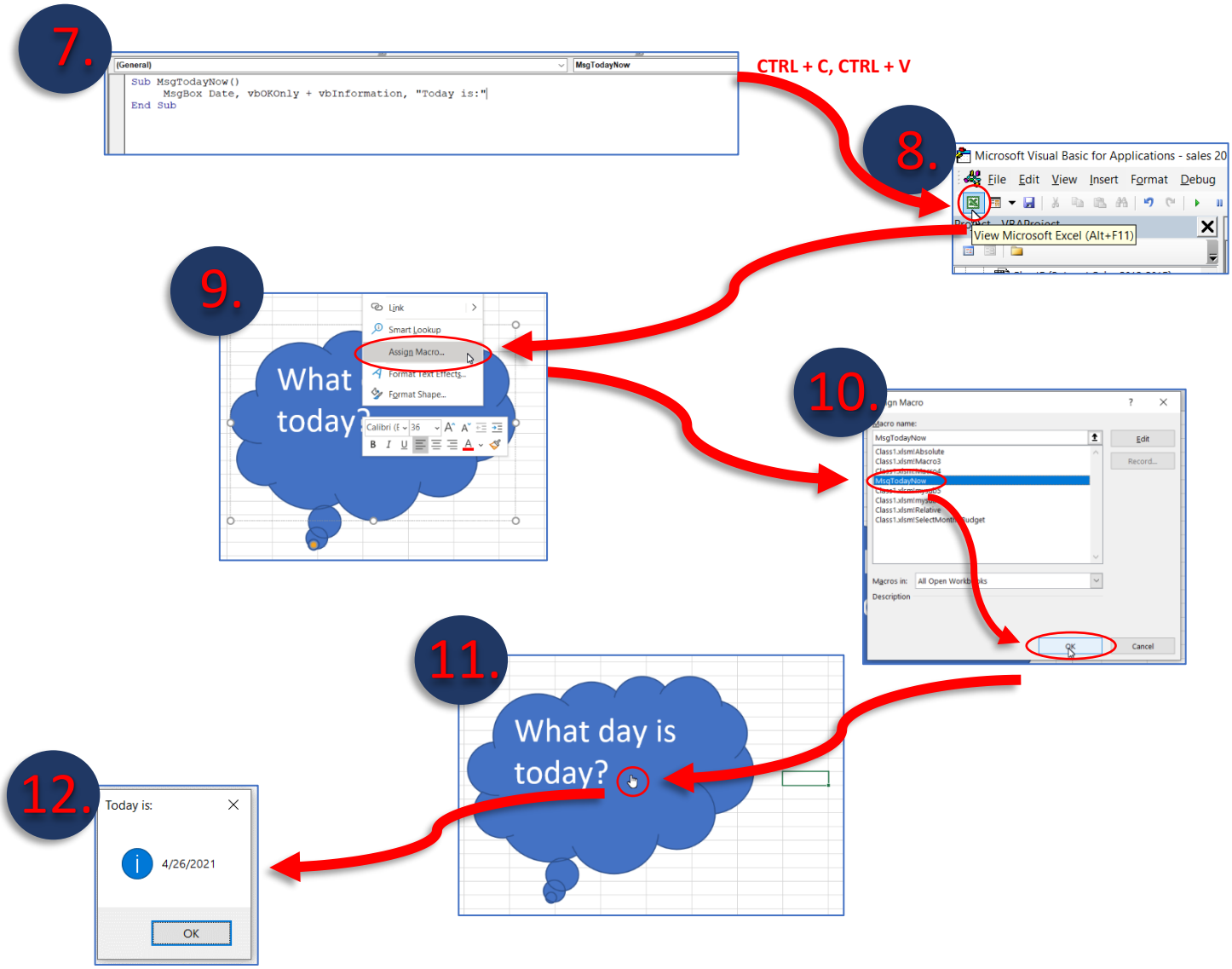


7. Type the code below into the new <Code Window>;

```
Sub MsgTodayNow()
    MsgBox Date, vbOKOnly + vbInformation, "Today is:"
End Sub
```

a. Pay attention to the name of the Procedure: <MsgTodayNow>

8. Go back to the Excel by clicking in the Excel Icon on the toolbar.;
9. Back in Excel, right-click in the shape and select <Assign Macro…>;
10. A Window with all available macros will open, choose the desired macro (MsgTodayNow) and click <OK>;
11. Click on any cell to unselect the shape;
12. The shape is now a button and when clicked will perform the actions determined in the macro when clicked: show the message with the current date and time. As you can see at the <Sheet 3> of <sales – 2013-2015> file.



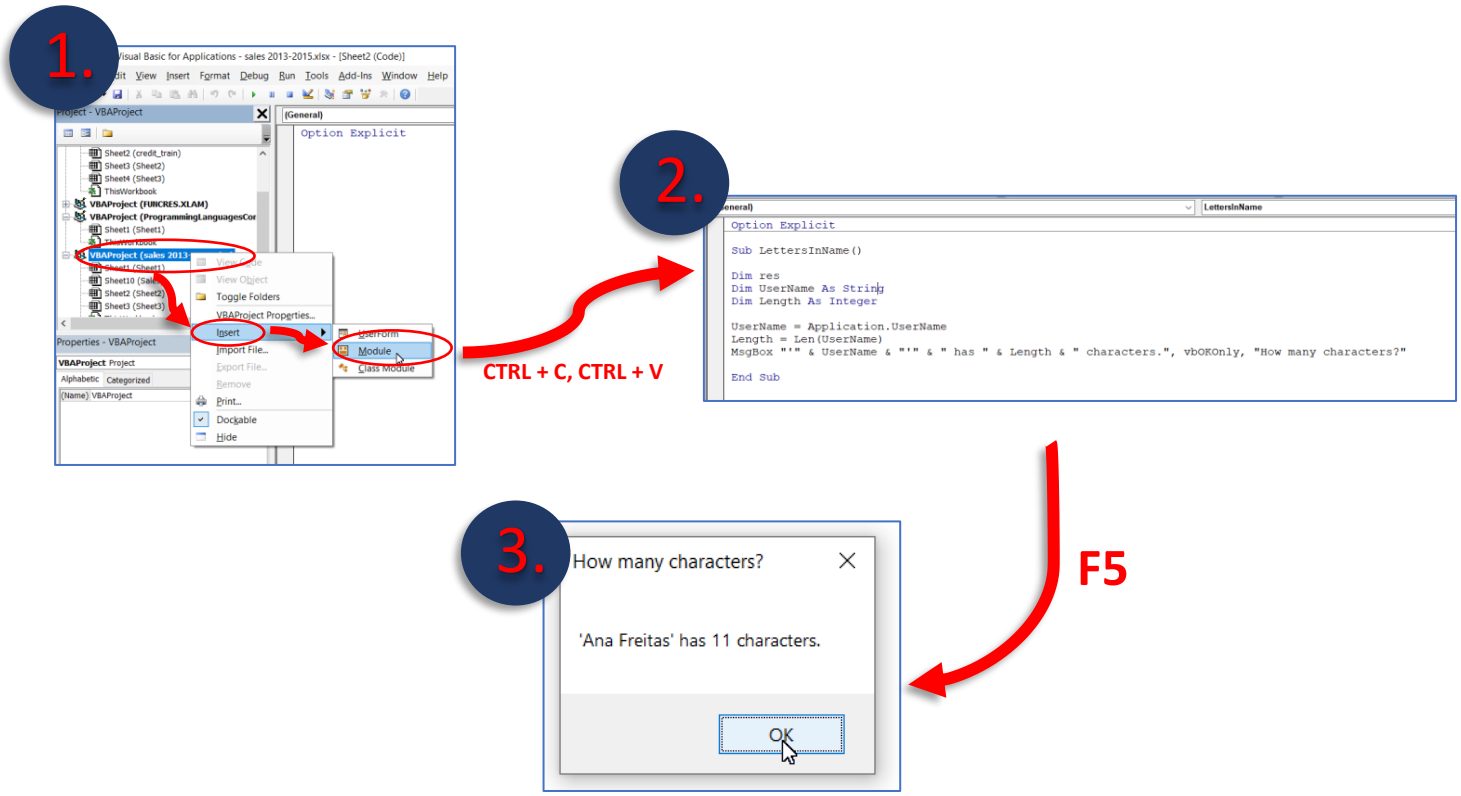
4. Create a procedure that declares variables of type string and integer.

With VBE opened:

1. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[1];
2. Type the code below into the new <Code Window>;

```
Option Explicit

Sub LettersInName()

Dim res
Dim UserName As String
Dim Length As Integer

UserName = Application.UserName
Length = Len(UserName) 'It returns the count of characters in the String ( spaces are counted)
MsgBox "'" & UserName & "'" & " has " & Length & " characters.", vbOKOnly, "How many characters?"

End Sub
```

3. Your procedure can be executed by pressing <F5> while the cursor is within the procedure. As you can see at the module <Letters> of <sales – 2013-2015> file. [2]



[1] You can, also, include the new procedure in an existent module.

[2] You can rename a module by altering its property at the <Properties window>.

Phase 2

1. Create a VBA procedure that changes the font color of table to bold.

1. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[3];
2. Type the code below into the new <Code Window>;

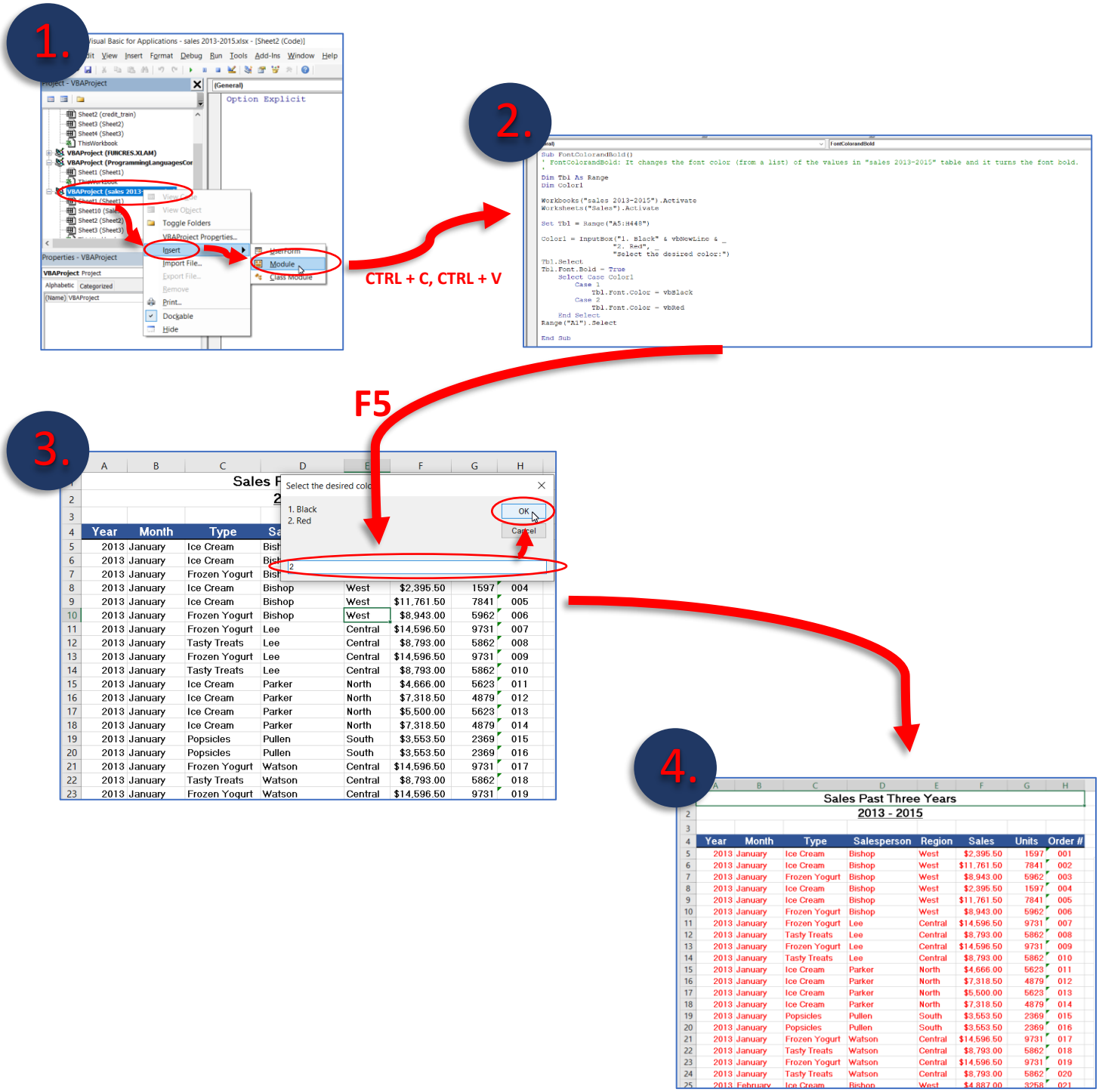
```
Sub FontColorandBold()
'
' FontColorandBold: It changes the font color (from a list) of the values in "sales 2013-2015" table and it turns the font bold.
'
Dim Tbl As Range
Dim Color1

Workbooks("sales 2013-2015").Activate
Worksheets("Sales").Activate

Set Tbl = Range("A5:H448")

Color1 = InputBox("1. Black" & vbNewLine & _
            "2. Red", _
            "Select the desired color:")
Tbl.Select
Tbl.Font.Bold = True
  Select Case Color1 'Colors the Font accordingly to user's selection
    Case 1
      Tbl.Font.Color = vbBlack
    Case 2
      Tbl.Font.Color = vbRed
  End Select
Range("A1").Select

End Sub
```

3. Your procedure can be executed by pressing <F5> while the cursor is within the procedure;
4. Input the desired color and click <OK>.
5. The programmed changes will be applied.

You can see the code at the module <Explanation> of <sales – 2013-2015> file. [4]

[3] You can, also, include the new procedure in an existent module.

[4] You can rename a module by altering its property at the <Properties window>.


1.
Visual Basic for Applications - sales 2013-2015.xlsx - [Sheet2 (Code)]
Insert
Module
Option Explicit
2.
Sub FontColorandBold()
' FontColorandBold: It changes the font color (from a list) of the values in "sales 2013-2015" table and it turns the font bold.
Dim Tbl As Range
Dim Color1
Workbooks("sales 2013-2015").Activate
Worksheets("Sales").Activate
Set Tbl = Range("A5:H448")
Color1 = InputBox("1. Black" & vbNewLine & _
"2. Red", , _
"Select the desired color:")
Tbl.Select
Tbl.Font.Bold = True
Select Case Color1
Case 1
Tbl.Font.Color = vbBlack
Case 2
Tbl.Font.Color = vbRed
End Select
Range("A1").Select
End Sub
CTRL + C, CTRL + V
F5
3.
Select the desired color
1. Black
2. Red
OK
Cancel
4.
Sales Past Three Years
2013 - 2015

2. Create a VBA procedure that adds a yellow explanation column to right of table. (the header of that column is explanation and fill that column by yellow).

1. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[5];
2. Type the code below into the new <Code Window>;

```
Sub AddExplanation()
'
'Determining the impact of a Sale (Explanation)
'Sales < 1000: Minor sale;
'Sales >= 1000: Good sale
'Sales < 5000: Great sale

Dim fRow As Long
Dim lRow As Long
Dim fColumn As Long
Dim lColumn As Long
Dim HFColor
Dim HFFont
Dim HFFill
Dim Cell1 As Range
Dim Rng As Range
Dim C As Range

Workbooks("sales 2013-2015").Activate
Worksheets("Sales").Activate

'Determining Table - Expects a cell reference within the table to be added the Explanation color
On Error GoTo Tbl1
Tbl1: Set Cell1 = Application.InputBox(Prompt:="Please select a cell within your data", Title:=" Where's your table?", Type:=8)
Cell1.Activate

Selection.End(xlUp).Select 'Identifies the first row and last column of header to extract its format
Selection.End(xlToRight).Select
ActiveCell.Select

'Retrieving Format of the existent header
HFFill = ActiveCell.Interior.ColorIndex
HFColor = ActiveCell.Font.ColorIndex
HFFont = ActiveCell.Font.FontStyle

'Finding the first blank cell at the side of header for additional column
fRow = ActiveCell.Row + 1
fColumn = ActiveCell.Column + 1

'Finding the last non-blank cell in column
lRow = ActiveCell.End(xlDown).Row
```

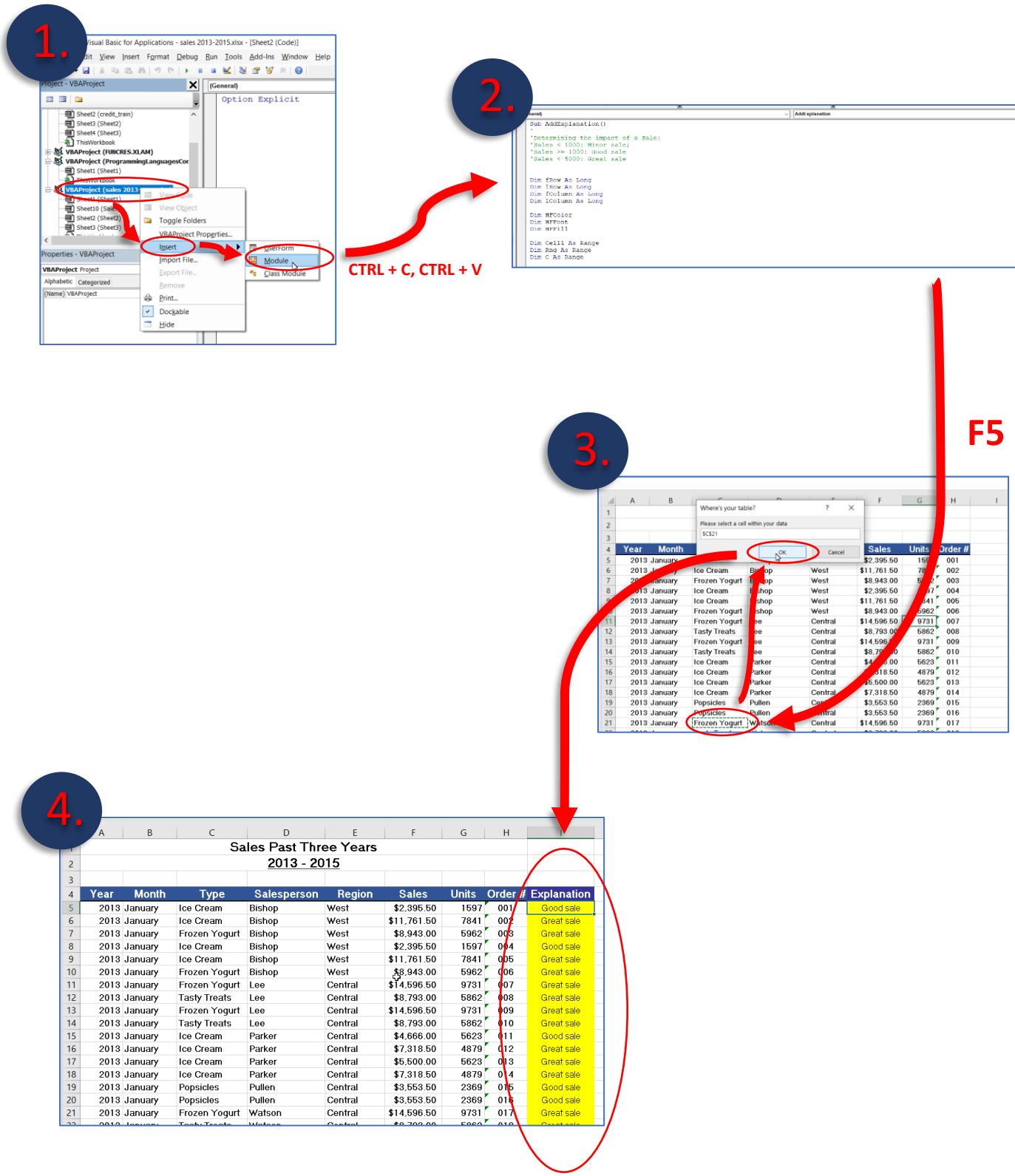
[5] You can, also, include the new procedure in an existent module.

```
'Adding and formatting Additional Column Header
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "Explanation"

With ActiveCell
   .Font.ColorIndex = HFColor
   .Font.FontStyle = HFFont
   .Interior.ColorIndex = HFFill
   l.Borders(xlEdgeBottom).LineStyle = xlContinuous
   .Borders(xlEdgeBottom).Weight = xlThin
   .EntireColumn.AutoFit
End With
'Identifying and Naming Sales column, storing its range
Range(Selection, Selection.End(xlToLeft)).Select

For Each C In Selection
   If C.Value = "Sales" Then
   C.Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Name = "Sales"
   End If
Next C

Selection.End(xlToRight).Select
ActiveCell.Select

'Inserting formula and Formatting Explanation's values
ActiveCell.Offset(1, 0).Select

ActiveCell.FormulaR1C1 = "=IF(Sales<1000,""Minor sale"",IF(AND(Sales>=1000,Sales<5000),""Good sale"",""Great sale""))"

ActiveCell.Interior.Color = vbYellow
ActiveCell.Select
Selection.NumberFormat = "General"

’Analysis of dataset shows that there is no blank cells making AutoFill possible and more efficient code
lColumn = ActiveCell.Column
Selection.AutoFill Destination:=Range(Cells(fRow, fColumn), Cells(lRow, lColumn))

End Sub
```

3. Your procedure can be executed by pressing <F5> while the cursor is within the procedure;

4. Your macro will apply the programmed steps. You can see the code at the module <Explanation> of <sales – 2013-2015> file. [6]

[6] You can rename a module by altering its property at the <Properties window>.


1.
Visual Basic for Applications - sales 2013-2015.xlsx - [Sheet2 (Code)]
Project - VBAProject
Option Explicit
Insert
Module
Class Module
User Form
Toggle Folders
VBAProject Properties...
Import File...
Export File...
Print...
Dockable
Hide
CTRL + C, CTRL + V
2.
Sub AddExplanation()
'Determining the impact of a Sale:
'Sales < 1000: Minor sale;
'Sales >= 1000: Good sale
'Sales < 5000: Great sale
F5
3.
Where's your table?
Please select a cell within your data
C21
OK
Cancel
4.
Sales Past Three Years
2013 - 2015
Year
Month
Type
Salesperson
Region
Sales
Units
Order #
Explanation

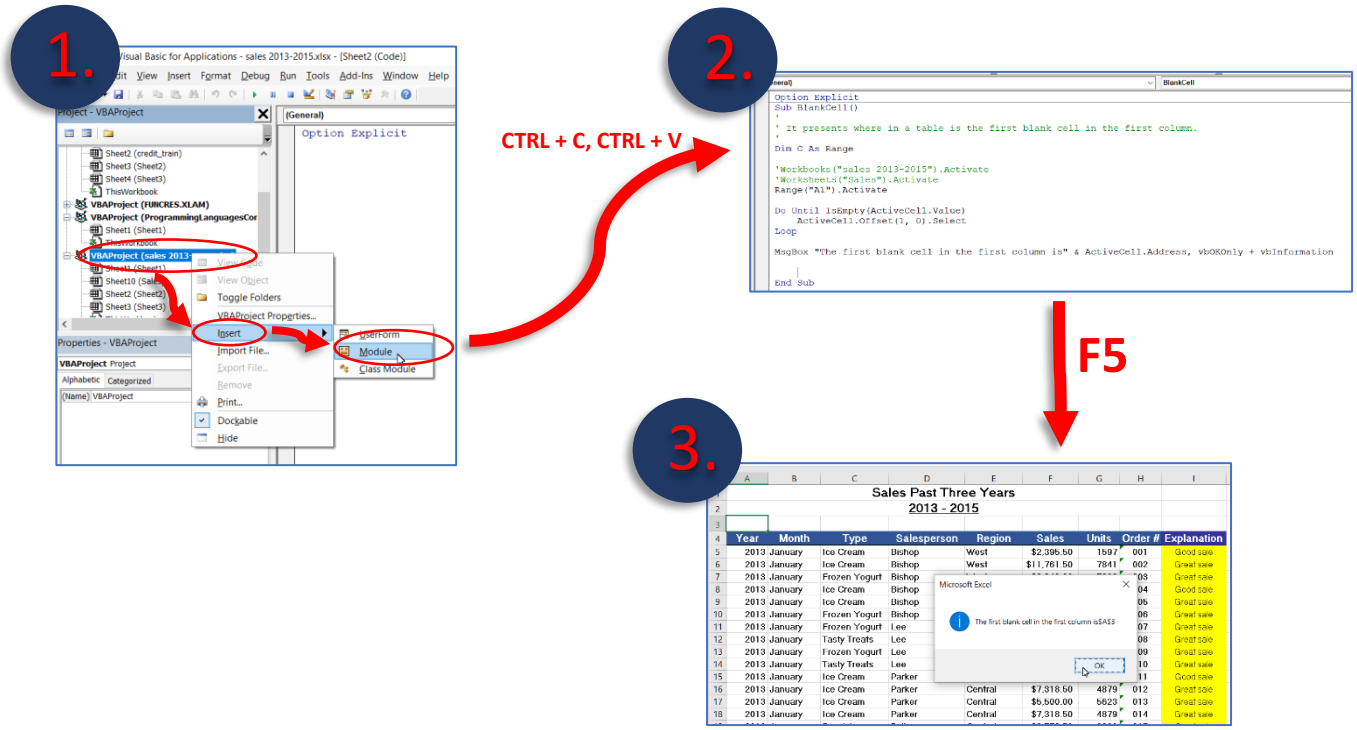
3. Create a VBA procedure that uses loop.

1. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[7];
2. Type the code below into the new <Code Window>;

```
Sub BlankCell()
' It presents where in the active worksheet is the first blank cell in the first column.
'
Dim C As Range

'Workbooks("sales 2013-2015").Activate
'Worksheets("Sales").Activate
Range("A1").Activate

Do until IsEmpty(ActiveCell.Value)
   ActiveCell.Offset(1, 0).Select
Loop

MsgBox "The first blank cell in the first column is" & ActiveCell.Address, vbOKOnly + vbInformation

End Sub
```

Do-Until Loop: it continues until the condition is met.
E.g.: An empty cell is met.

3. Your procedure can be executed by pressing <F5> while the cursor is within the procedure. You can see the code at the module <Module4> of <sales – 2013-2015> file. [8]



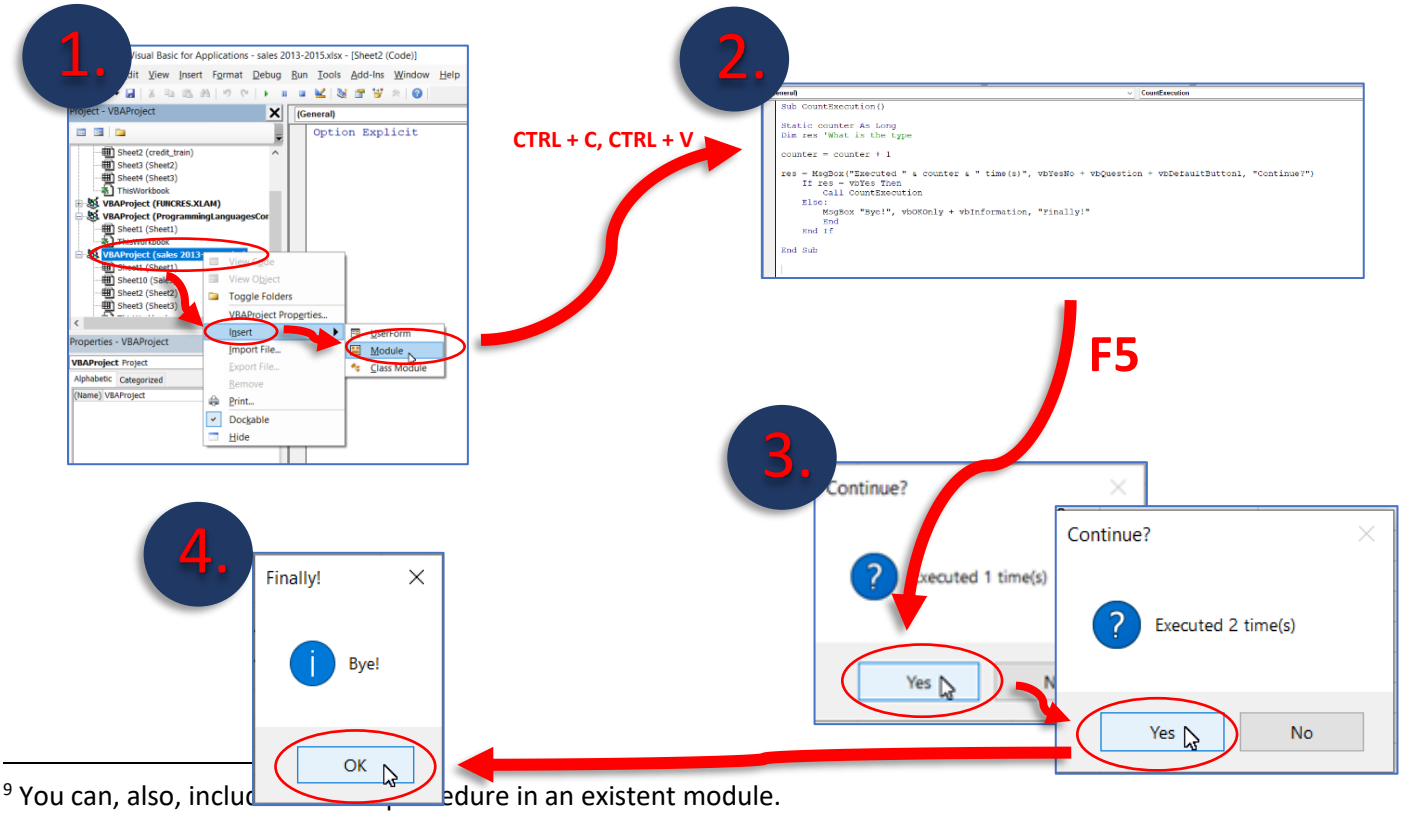
[7] You can, also, include the new procedure in an existent module.
[8] You can rename a module by altering its property at the <Properties window>.

4. Create a message box that displays number of executions of one procedure.

1. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[9];
2. Type the code below into the new <Code Window>;

```
Sub CountExecution()

Static counter As Long ‘ it retains the value of variable when the procedure ends
Dim res 'What is the type

counter = counter + 1 ‘it stores the number of executions

res = MsgBox("Executed " & counter & " time(s)", vbYesNo + vbQuestion + vbDefaultButton1, "Continue?")
   If res = vbYes Then
      Call CountExecution
   Else:
      MsgBox "Bye!", vbOKOnly + vbInformation , "Finally!"
      End
   End If

End Sub
```

3. Insert a Your procedure can be executed by pressing <F5> while the cursor is within the procedure. You can see the code at the module <Module4> of <sales – 2013-2015> file. [10]



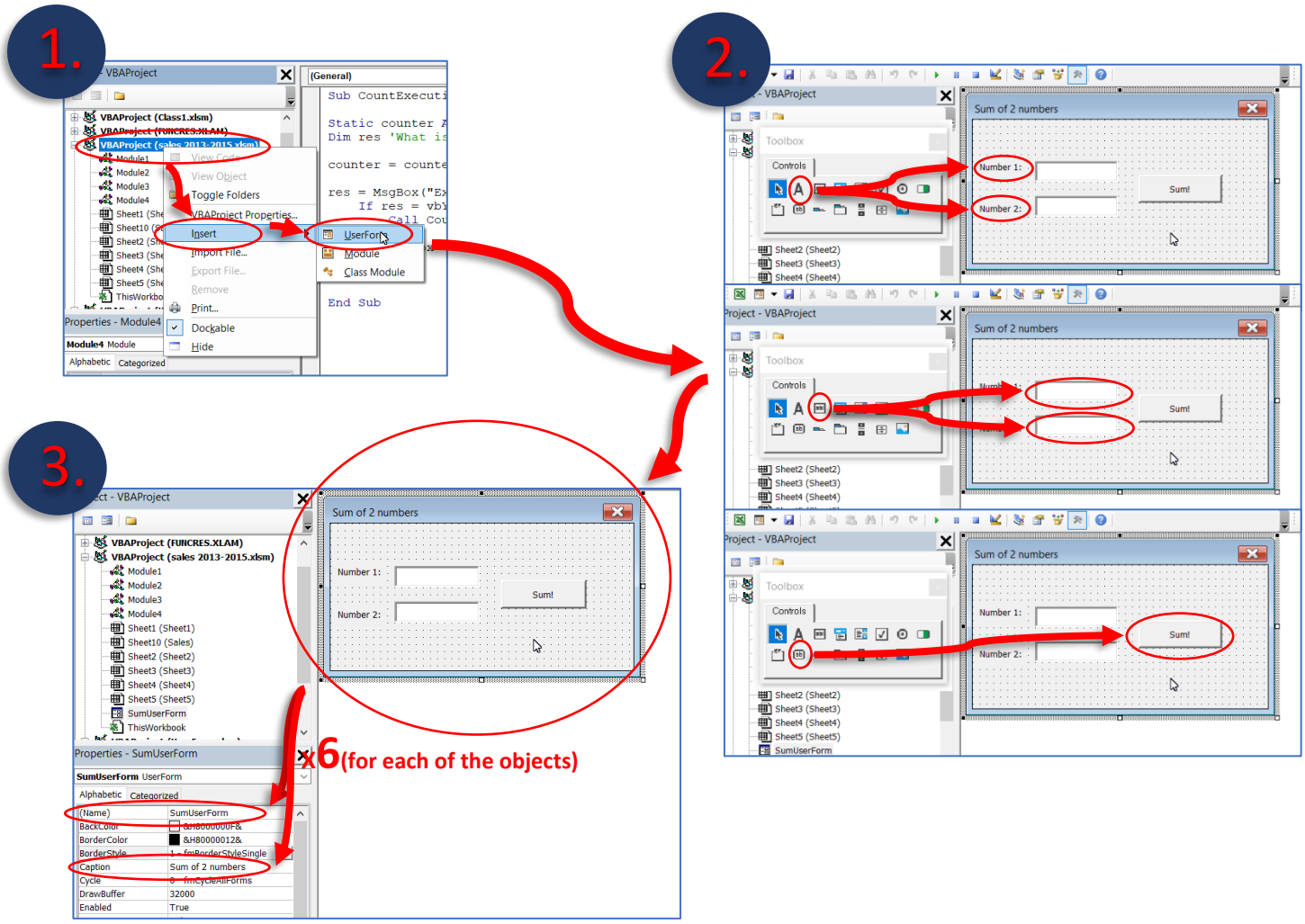
[9] You can, also, inclu… …edure in an existent module.

[10] You can rename a module by altering its property at the <Properties window>.

5. Create a user Form that has two text boxes and a button to calculate sum and show it in a message box.

1. Insert a new UserForm by right-clicking in your workbook name in the <Project window>, then select <Insert> and <UserForm>. A new <UserForm> will open.;
2. Insert by clicking and dragging the required elements: 2 text boxes and the button. Optionally, you may add labels to the text boxes;
3. Format (Select the object and change the property in the <Properties Window> each of the inserted elements with the <Name> and <Caption>[11][12] below:

Element	Name	Caption
User Form	SumUserForm	Sum of 2 numbers
Label 1	Number1Label	Number 1:
Label 2	Number2Label	Number 2:
Text Box 1	TextBox1	N/A
Text Box 2	TextBox2	N/A
Command Button	SumButton	Sum!



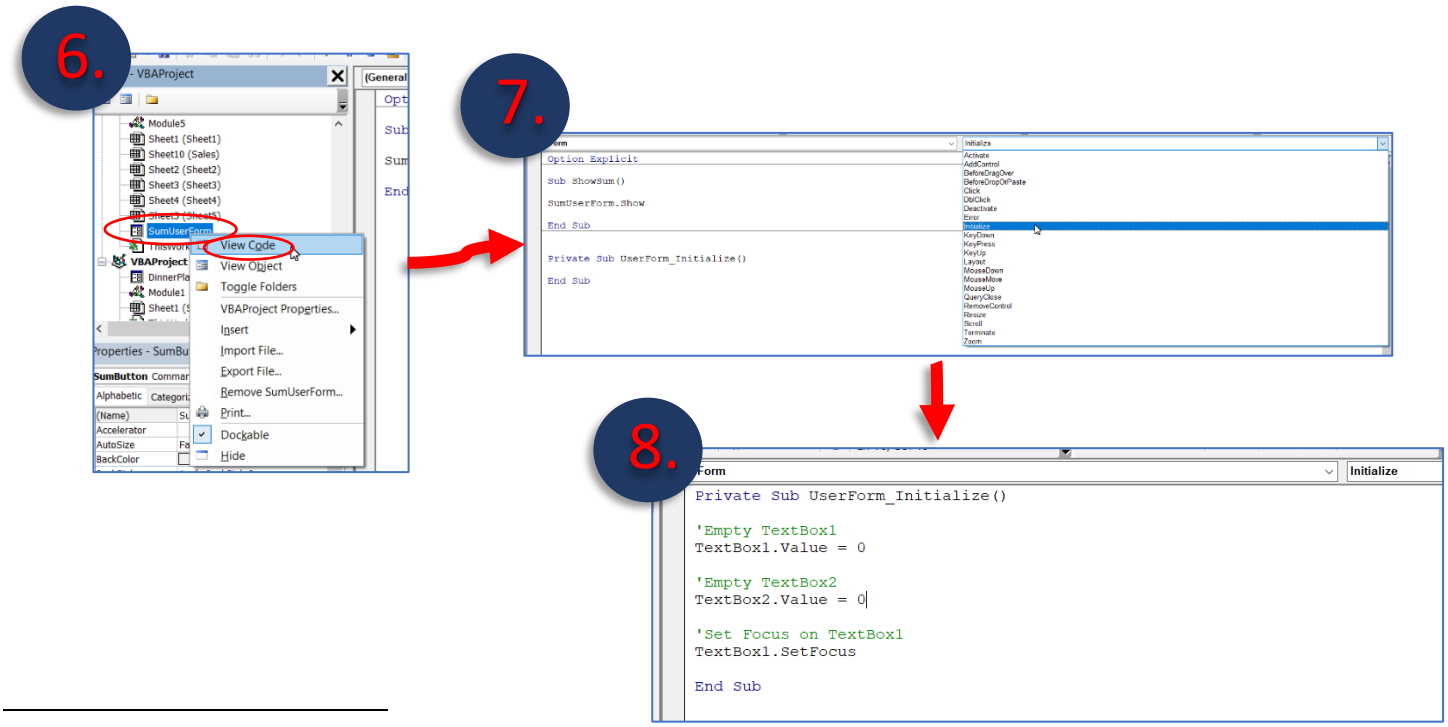
[12] When applicable.

4. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module (<Module5>) will open[13];
5. Type the code below into the new <Code Window>. Remember the name of this procedure: <ShowSum>;

```
Sub ShowSum() 'Macro to call the form
SumUserForm.Show
End Sub
```

6. Right-Click <SumUserForm> then select <View Code>;
7. Choose from the right drop-down list the object <UserForm> and from the left drop-down, the event <Initialize>;
8. Type the code below into the new <Code Window>;

```
Private Sub UserForm_Initialize() 'Macro to Initialize properly the form

'Empty TextBox1
TextBox1.Value = 0

'Empty TextBox2
TextBox2.Value = 0

'Set Focus on TextBox1
TextBox1.SetFocus

End Sub
```



[13] You can, also, include the new procedure in an existent module.

9. Choose from the right drop-down list the object <SumButton> and from the left drop-down, the event <Click> and type the code below into the new <Code Window>;

Private Sub SumButton_Click() 'Macro to determine the actions to be performed when SumButton is clicked

Dim Sum1 As Double

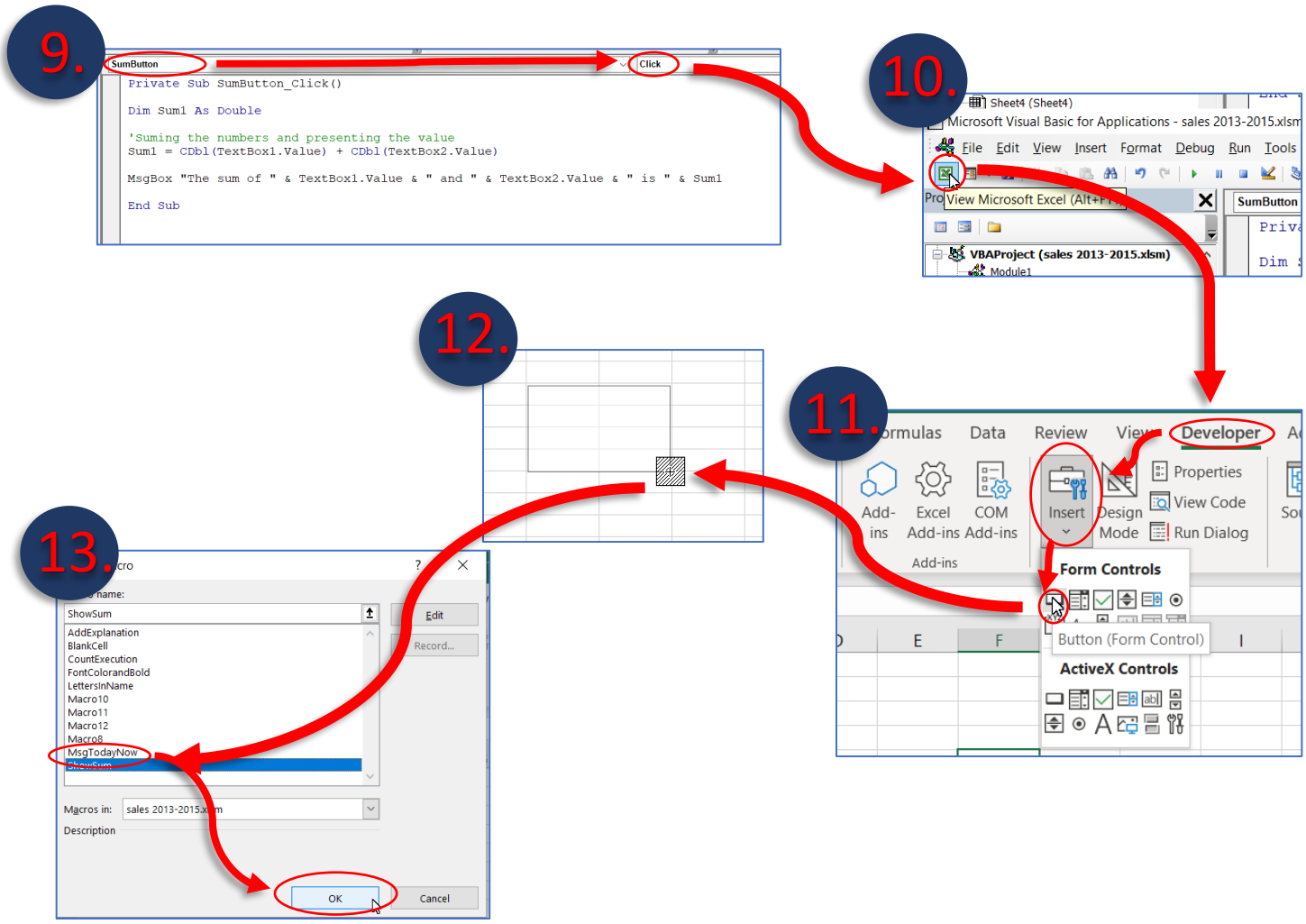
'Suming the numbers and presenting the value
Sum1 = CDbl(TextBox1.Value) + CDbl(TextBox2.Value)

MsgBox "The sum of " & TextBox1.Value & " and " & TextBox2.Value & " is " & Sum1

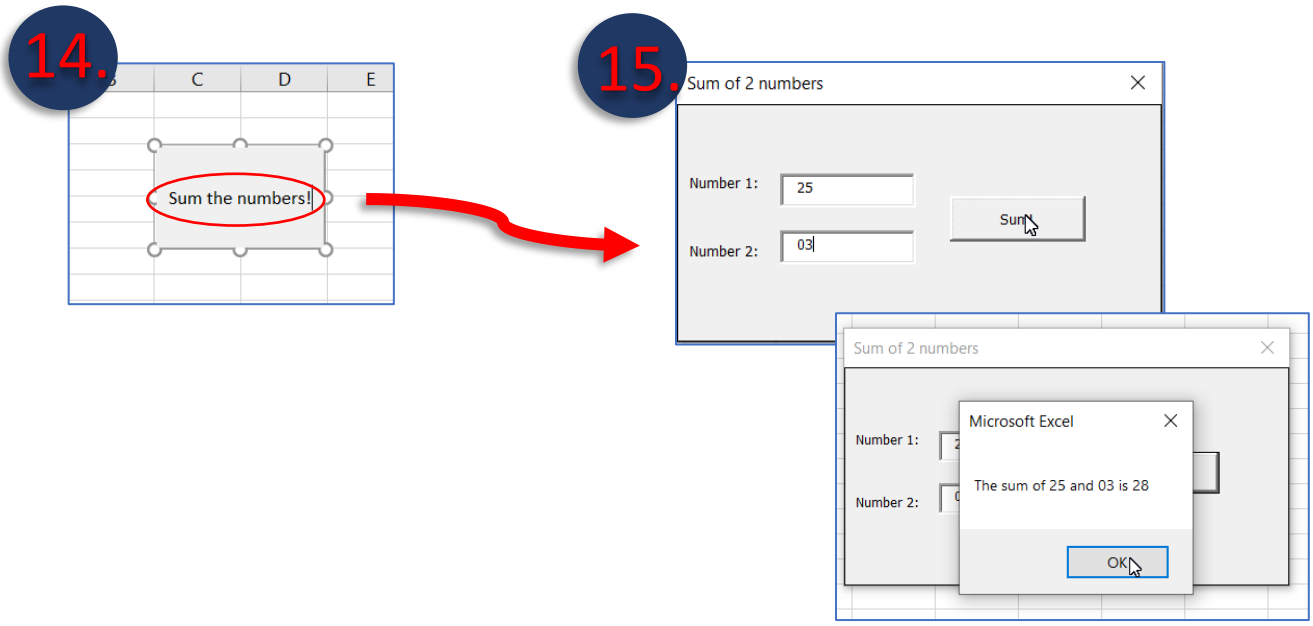
End Sub

---

10. Go back to the worksheet, by clicking the <View Microsoft Excel> button;
11. Go to the <Developer> menu, <Controls> area and click <Insert> button;
12. Insert the new button by clicking and dragging in the worksheet until the desired size is met;
13. A window will open. Assign the macro <ShowSum>.;



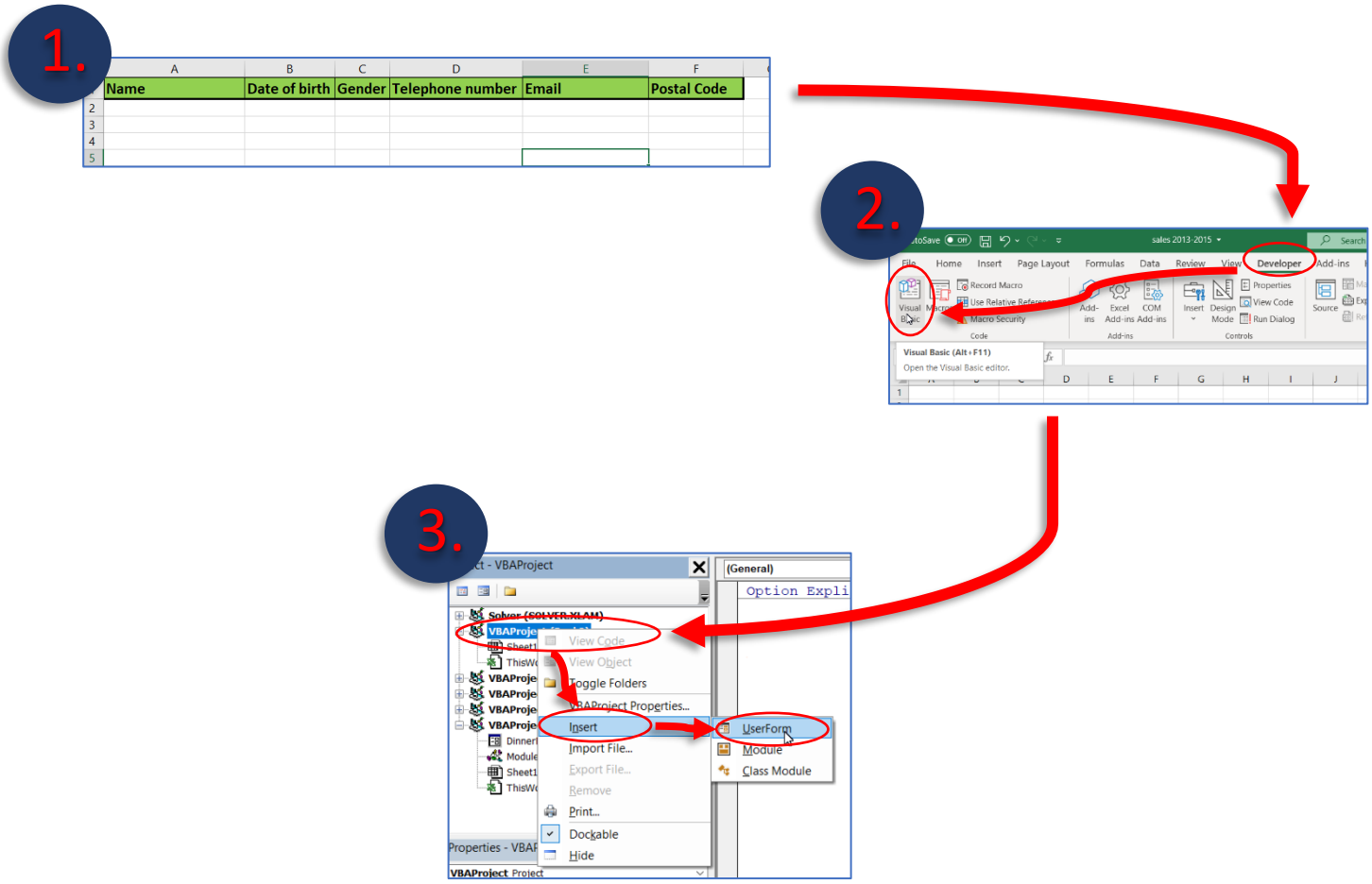
14. Insert the label <Sum the numbers!> and click anywhere else in the worksheet. Your button is ready.
15. Just click the button and your macro will call the form.



6. Create a User Form in Excel VBA to get name, date of birth, gender, telephone number, email, and postal code from the user and store the value provided by the user in the worksheet

With a new workbook opened:

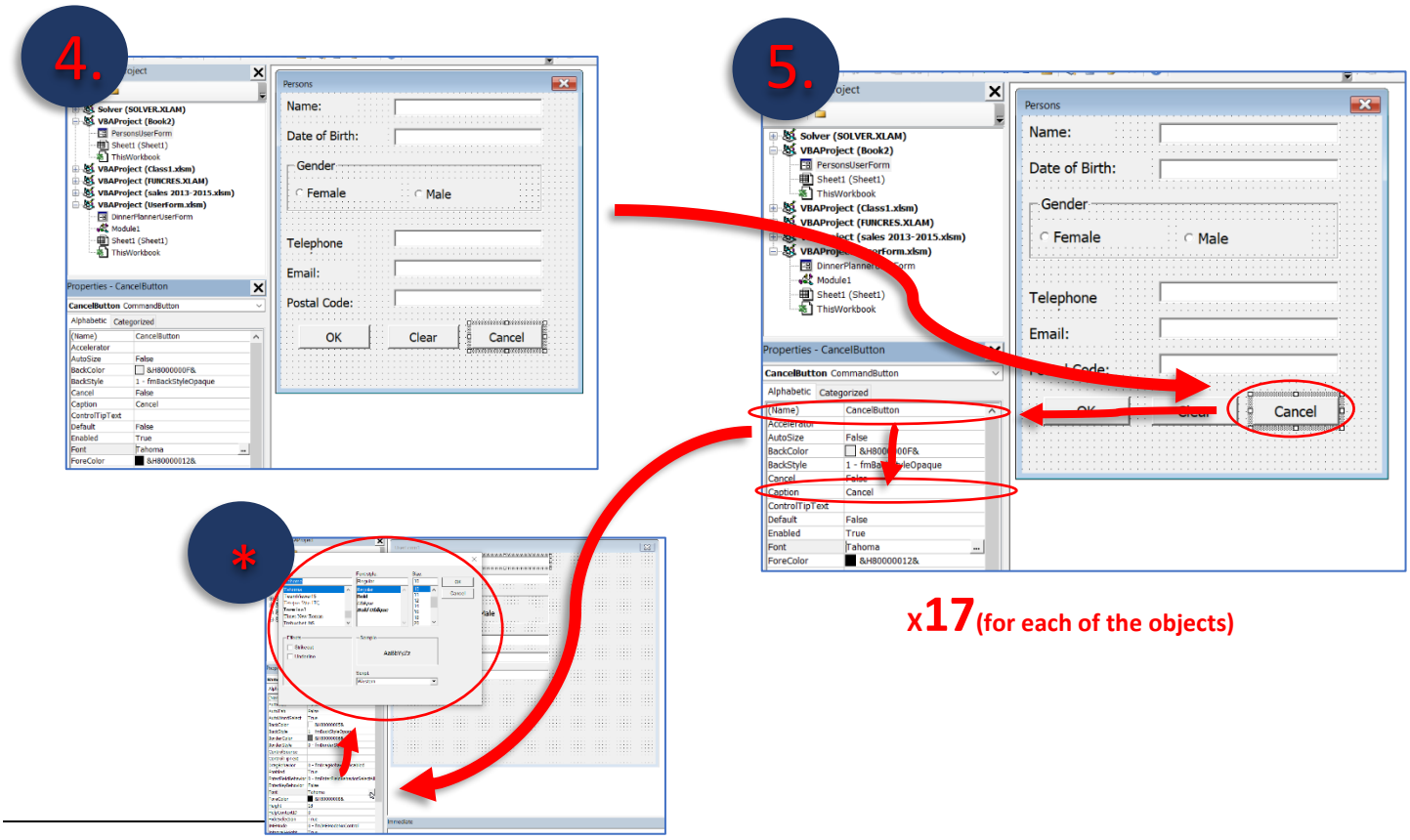
1. Insert into the first row the required headers: name, date of birth, gender, telephone number, email, and postal code. Format it as desired;
2. Go to The VBE (<Developer> menu, then <Visual Basic> in the <Code area>);
3. Insert a new <User Form> by right-clicking in your workbook name in the <Project window>, then select <Insert> and <UserForm >. A new <UserForm> will open.;



4. Insert by clicking and dragging the required elements: 5 labels, 5 text boxes, 1 frame, 2 option buttons, and 3 command buttons.
5. Format each of the inserted elements with the <Name> and <Caption>[14] as below:

Element	Name	Caption
User Form	PersonsUserForm	Persons
Label 1	NameLabel	Name:
Label 2	DateLabel	Date of Birth:
Label 3	TelephoneLabel	Telephone number:
Label 4	EmailLabel	Email:
Label 5	PostalLabel	Postal Code:
Text Box 1	NameTextBox	N/A
Text Box 2	DateTextBox	N/A
Text Box 3	TelephoneTextBox	N/A
Text Box 4	EmailTextBox	N/A
Text Box 5	PostalTextBox	N/A
Frame	GenderFrame	Gender
Option Button 1	FemaleOptionButton	Female
Option Button 2	MaleOptionButton	Male
Command Button 1	OKButton	OK
Command Button 2	ClearButton	Clear
Command Button 3	CancelButton	Cancel

* You can adjust the font to provide better view by changing the font property of each of the objects.

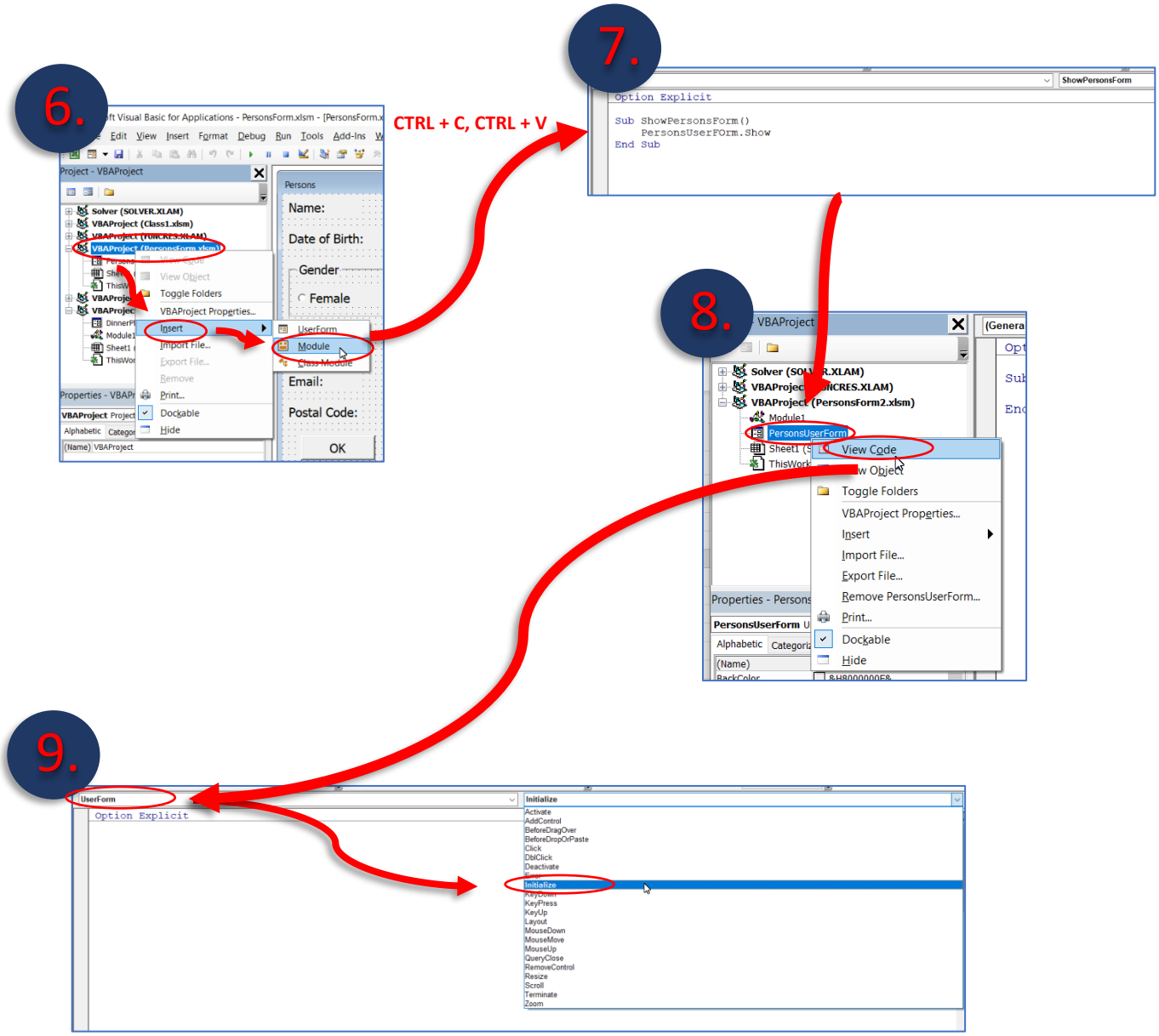


[14] When applicable.

6. Insert a new module by right-clicking in your workbook name in the <Project window>, then select <Insert> and <Module>. A new <Code Window> for this new module will open[15];
7. Type the code below into the new <Code Window>. Remember the name of this procedure: <ShowPersonsForm>;

```
Sub ShowPersonsForm()
    PersonsUserFOrm.Show
End Sub
```

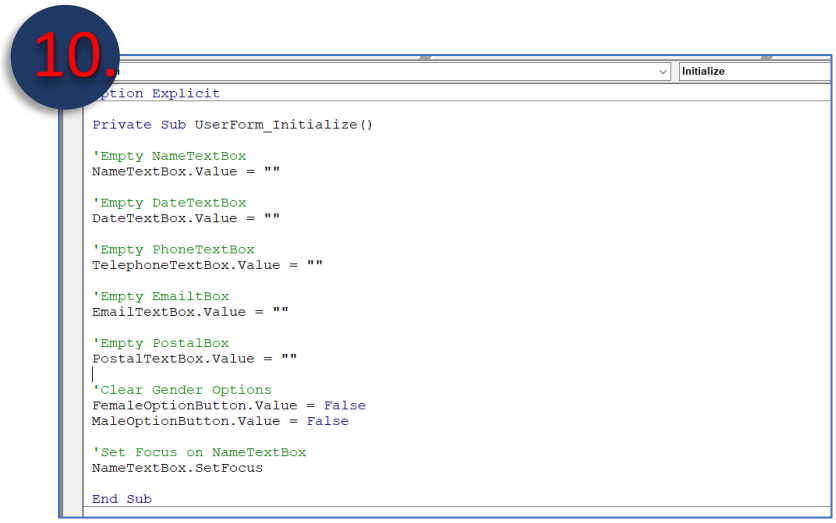
8. Right-Click <PersonsUserForm> then select <View Code>;
9. Choose from the right drop-down list the object <UserForm> and from the left drop-down, the event <Initialize>;



[15] You can, also, include the new procedure in an existent module.

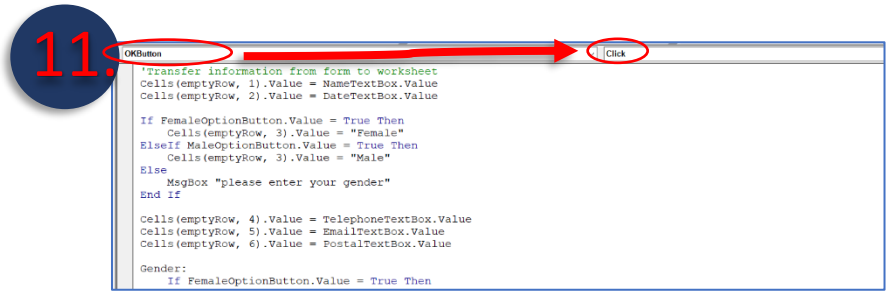
10. Type the code below into the new <Code Window:

```
'Empty NameTextBox
NameTextBox.Value = ""

'Empty DateTextBox
DateTextBox.Value = ""

'Empty PhoneTextBox
TelephoneTextBox.Value = ""

'Empty EmailtBox
EmailTextBox.Value = ""

'Empty PostalBox
PostalTextBox.Value = ""

'Clear Gender Options
FemaleOptionButton.Value = False
MaleOptionButton.Value = False

'Set Focus on NameTextBox
NameTextBox.SetFocus
```



11. Choose from the right drop-down list the object <OKButton> and from the left drop-down, the event <Click> and Type the code below into the new <Code Window:

```
If NameTextBox.Value = "" Then MsgBox "please enter your name"

Dim emptyRow As Long

'Make Sheet1 active
Sheet1.Activate

'Determine emptyRow
emptyRow = WorksheetFunction.CountA(Range("A:A")) + 1 'CountA - counts the number of non-blank values and adds one to empty row

'Gender informed previously?
If Cells(emptyRow - 1, 3).Value = "" Then GoTo Gender

'Transfer information from form to worksheet
Cells(emptyRow, 1).Value = NameTextBox.Value
Cells(emptyRow, 2).Value = DateTextBox.Value

If FemaleOptionButton.Value = True Then
   Cells(emptyRow, 3).Value = "Female"
ElseIf MaleOptionButton.Value = True Then
   Cells(emptyRow, 3).Value = "Male"
Else
   MsgBox "please enter your gender"
End If

Cells(emptyRow, 4).Value = TelephoneTextBox.Value
Cells(emptyRow, 5).Value = EmailTextBox.Value
Cells(emptyRow, 6).Value = PostalTextBox.Value

Gender:
   If FemaleOptionButton.Value = True Then
      Cells(emptyRow - 1, 3).Value = "Female"
   ElseIf MaleOptionButton.Value = True Then
      Cells(emptyRow - 1, 3).Value = "Male"
   End If
```

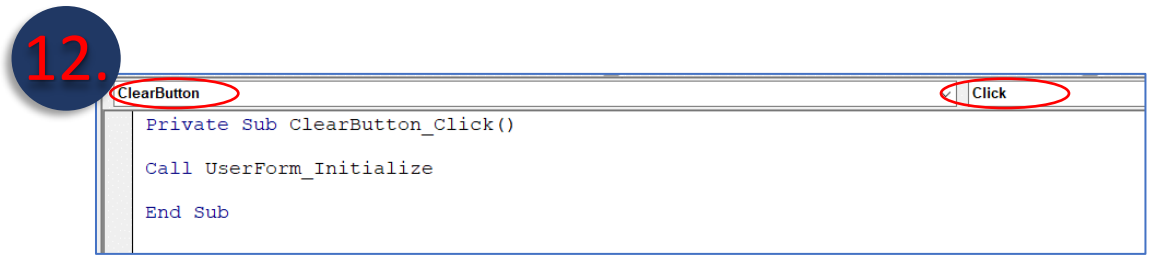


12. Choose from the right drop-down list the object <ClearButton> and from the left drop-down, the event <Click> and Type the code below into the new <Code Window:

Private Sub ClearButton_Click()

Call UserForm_Initialize

End Sub

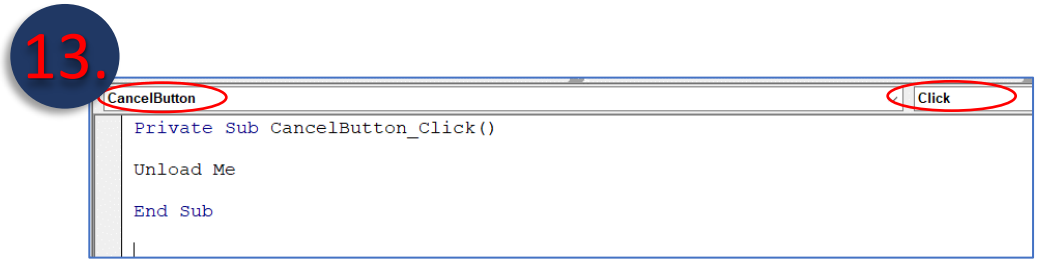


13. Choose from the right drop-down list the object <CancelButton> and from the left drop-down, the event <Click> and Type the code below into the new <Code Window:

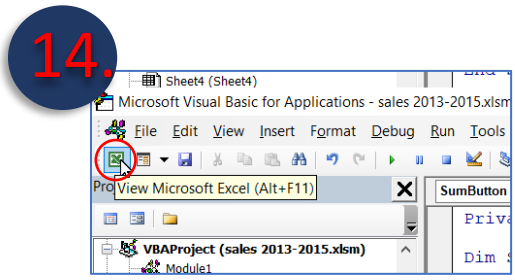
Private Sub CancelButton_Click()

Unload Me

End Sub

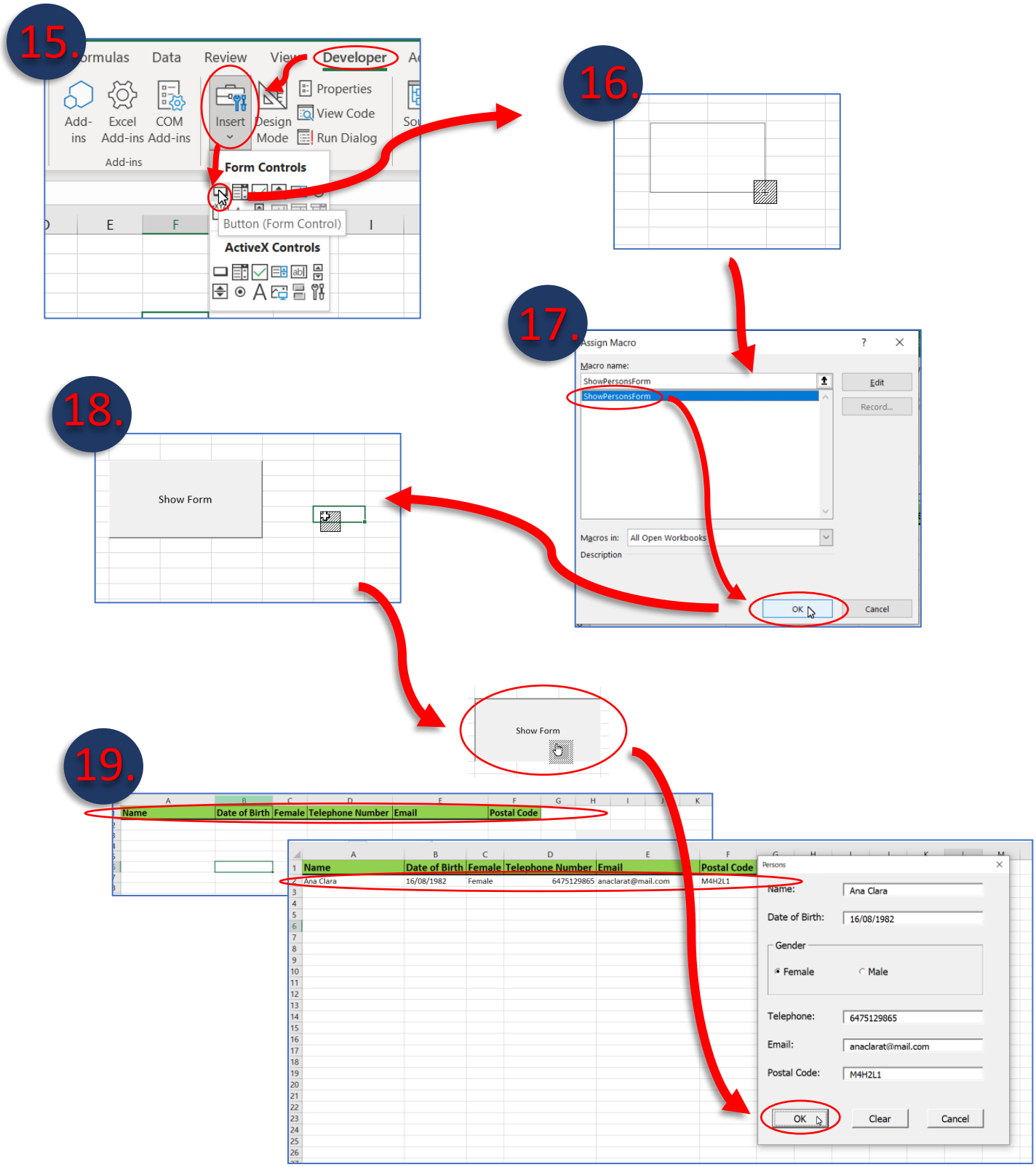


14. Go back to the worksheet, by clicking the <View Microsoft Excel> button;



15. Go to the <Developer> menu, <Controls> area and click <Insert> button;
16. Insert the new button by clicking and dragging in the worksheet until the desired size is met;
17. A window will open. Assign the macro <ShowPersonsForm>;
18. Insert the label <Show Form> and click anywhere else in the worksheet. Your button is ready.
19. Just click the button and your macro will call the form.

You can see this code at the <PersonsForm> file.



Conclusion

Excel and VBA are excellent tools to help you to perform exploratory Data Analysis, generate reports, and automate actions.

With little effort by recording a macro and edit when necessary, daily tasks can be done only once and be executed in future with just a click of a button, emphasizing the importance of its learning.