How to Excel Part 3

Finance Training

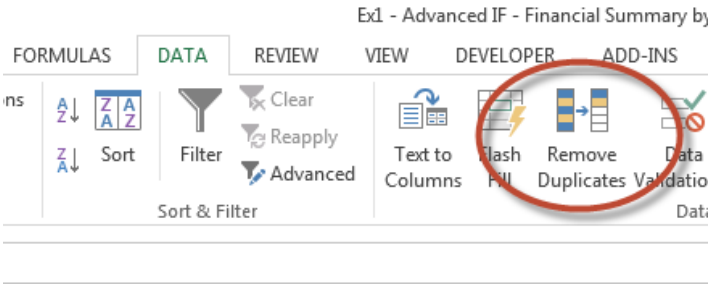
Contents

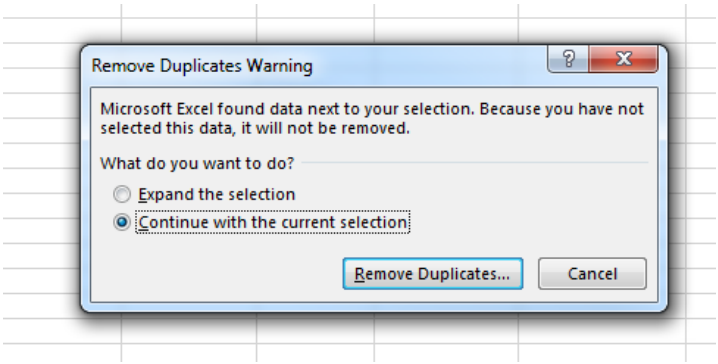
Exercise 1: Advanced IF formulas

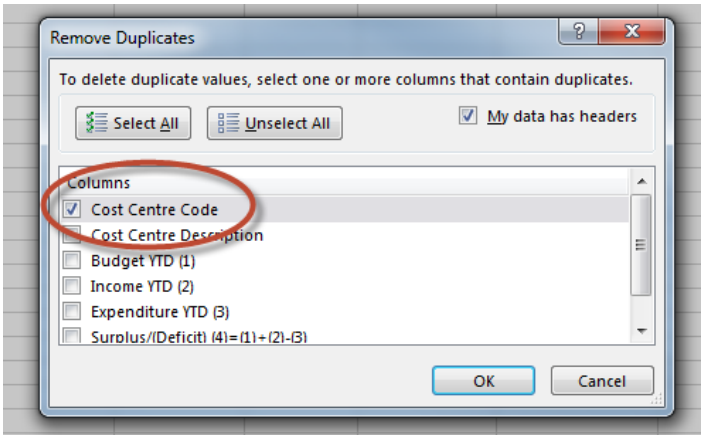
Open the data document **Ex1 Advanced IF – Financial Summary by Source of Funds**

(The tasks are also listed in the task tab of the spreadsheet)

1. Copy the list of **Cost Centres** to the analysis worksheet
2. Delete any duplicates







Select the column that you want to use as the basis for identifying duplicates

3. Use a VLOOKUP to insert the Cost Centre description from the source data page (Page1).

B2 =VLOOKUP(A2,Page1!$F$6:$G$258,2,FALSE)

	A	B	C
1	Cost Centre Code	Cost Centre Description	Budget YTD (1)
2	0000	Balance Sheet Default	
3	PDYA	Key Deposits	
4	PDYC	Private Purchases	
5	PDAA		
6	PDAG		
7	PDBA		
8	PDCB		
9	PDDB		

4. Use SUMIF formula to find the total of the entries against each unique Cost Centre (repeat for each columns C to E)

=SUMIF(Range,Criteria,[sum_range])

Range = Cost centre column in original data set

Criteria = The Cost Centre in the analysis sheet

Sum_range = The column in the original data set that needs to be added

E3 =SUMIF(Page1!$F$6:$F$258,Analysis!$A3,Page1!M$6:M$258)

	A	B	C	D	E	F	G
1	Cost Centre Code	Cost Centre Description	Budget YTD (1)	Income YTD (2)	Expenditure YTD (3)	Surplus/(Deficit) (4)=(1)+(2)-(3)	Number of SoF included in total
2	0000	Balance Sheet Default	0	0	0	0	
3	PDYA	Key Deposits	0	0	111.84	-111.84	
4	PDYC	Private Purchases	0	107.7	74.04	33.66	
5	PDAA						
6	PDAG						
7	PDBA						
8	PDCB						

NOTE – Absolute cell referencing

$A$6 = The cell used will not change when the formula is copied

$A6 = The column will not change but the row will when the formula is copied

A$6 = The row will remain static but the column will change when the formula is copied

Select the cell reference in the formula and use F4 to cycle round the various options

5. Calculate the surplus for each Cost Centre

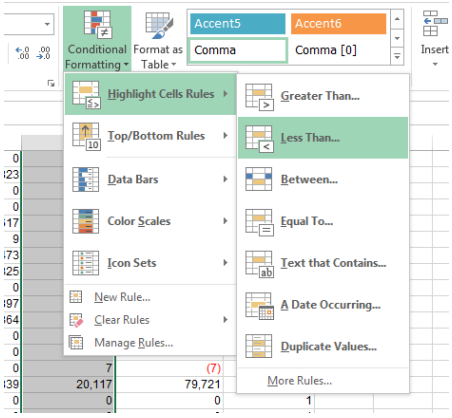
6. Use the COUNTIF formula to count the number of times the Cost Centre code appears in the list (i.e. the number of Source of Funds)

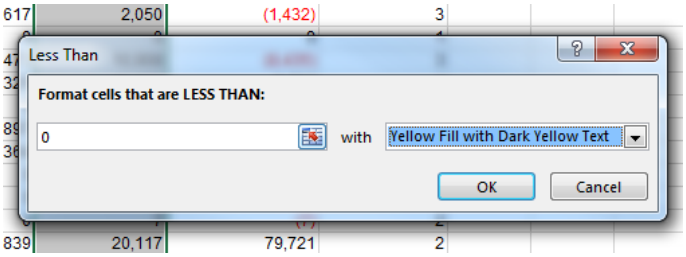
7. Total each of the columns on the Analysis worksheet

8. Create a check to ensure that the totals on the analysis sheet equal the totals in the original data

9. Format all the numbers to show the comma separator, no decimal points and negative values with brackets and in red

10. Use conditional formatting to highlight any negative expenditure in the expenditure column (highlight in yellow)





11. Use conditional formatting to highlight (in green) any surplus or deficit exceeding £30,000

Exercise 2: INDEX MATCH MATCH

A more advanced version of the LOOKUP functions. LOOKUP can only be used to look in rows or columns but not both.

	A	B	C
1		**Sales**	**Number of sales**
2	January	12,146	260,402
3	February	17,513	231,554
4	March	14,495	255,504
5	April	12,382	249,615
6	May	13,010	212,131
7	June	13,067	201,395
8	July	17,402	245,997
9	August	16,557	264,274
10	September	16,378	234,562
11	October	13,167	250,273
12	November	13,857	233,827
13	December	14,492	275,743

	K	L	M	N
1				
2		Month	October	250,273
3		Item	Number of sales	

This combines two functions

1. INDEX the whole table **INDEX(A1:C13)**
2. The first MATCH will find the required row, in the example above this is found by matching the month. **MATCH(M2,A1:A13,0)**
3. The second MATCH will find the required column by matching with the titles. **MATCH(M3,A1:C1,0)**

NB the 0 is just used to indicate that it should look for an exact match

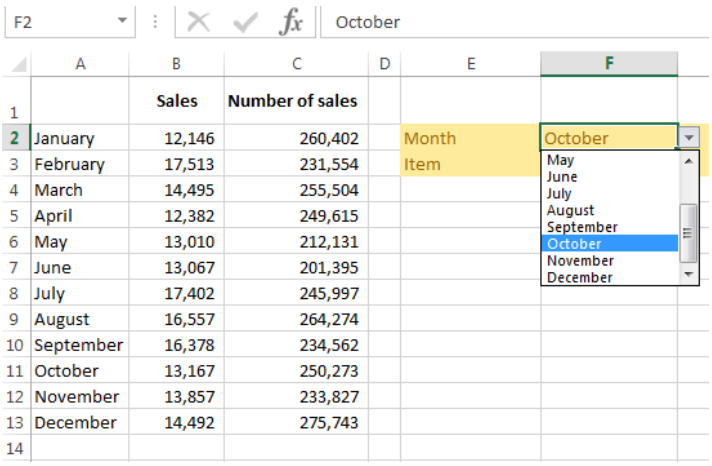
The complete formula is:

=INDEX(A1:C13,MATCH(M2,A1:A13,0),MATCH(M3,A1:C1,0))

In essence this will do the same as a VLOOKUP but are definite benefits and more that can be done:

- You don't have to count. With INDEX MATCH, there's no more worrying about counting to figure out which column you need to pull from. You just select your lookup column and your results column.
- You can safely insert columns. With VLOOKUP, if you insert a column in between the start of your table and the column you want to reference, your formula will break — the column_index_number within your VLOOKUP won't update. INDEX MATCH, on the other hand, safely updates no matter where you insert columns.
- You can lookup backwards. VLOOKUP only allows you to look up from columns that are in front of your starting point. With INDEX MATCH, you can pull from any column you want to.

Data validation



Use data validation to create drop down lists for the months and column headings

TASK

---

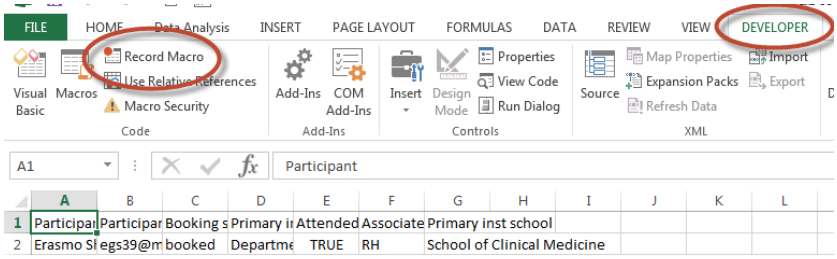
a) Open the document **Ex2 INDEX MATCH_task**
b) Enter formulas in the yellow boxes of the task sheet that displays the information required.

Exercise 3 – Recording Macros

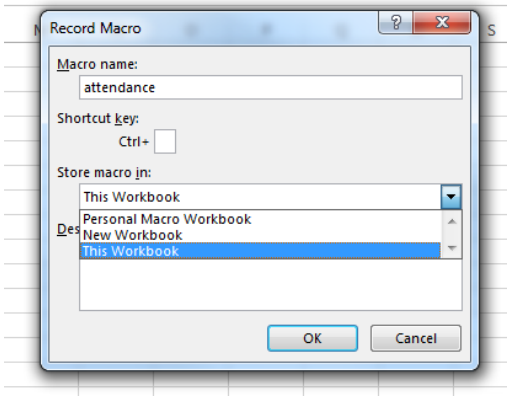
If you go to the **Developer tab** and select **Record Macro** anything that you do after that point will be recorded.

---

a) Open the workbook **Ex3 Source data**
b) Make sure the source data tab is selected (a fictional list of attendees on a training course)
c) Select **Record Macro**



d) Give your macro a name and indicate where it is to be stored.

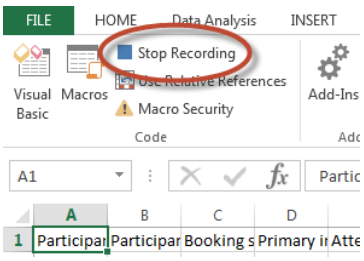


Name – *Attendance*

Store macro in *This workbook*

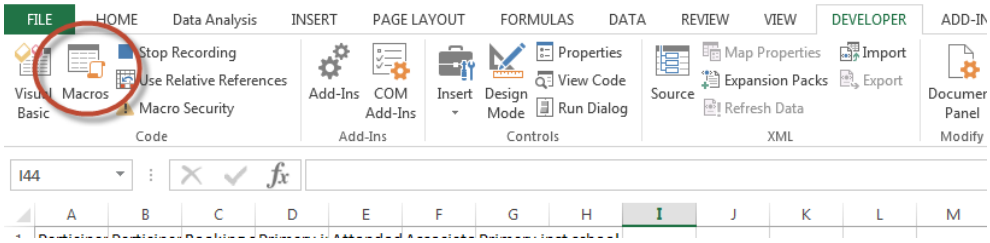
e) Make your recording including the following Tasks:
- Split the names into two columns and label Firstname and Surname
- Remove duplicates
- Delete booking status column
- Delete attended column
- Set column widths so that all of the data fits ?
- Make all the headings bold ?
- Rename the column *Associated inst CUFS code* heading to just **CUFS code**

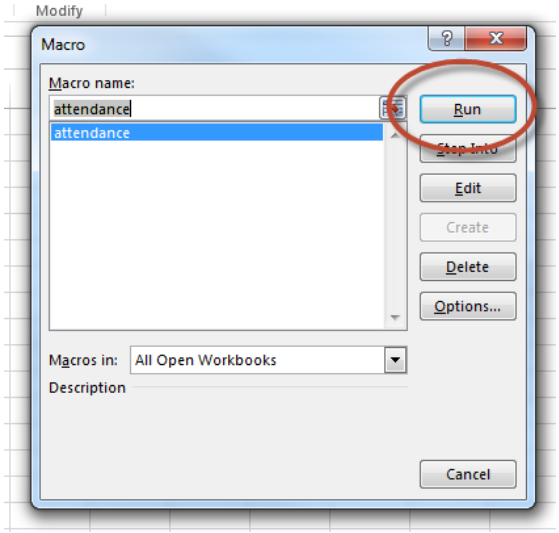
f) Stop recording



g) Test on the *Data to test macro* worksheet

- Developer tab
- Macros
- Select **Attendance** macro
- Run



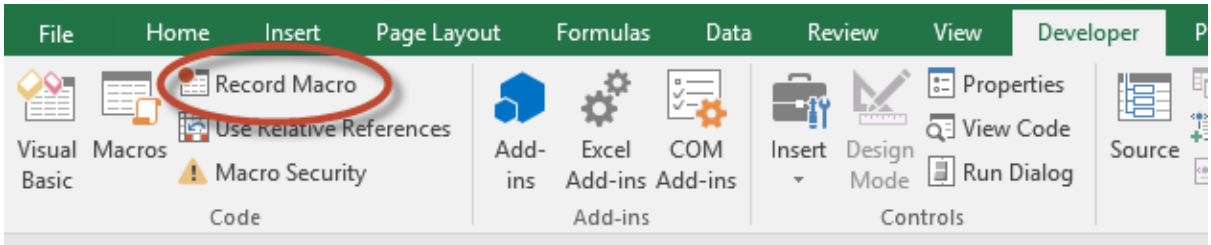


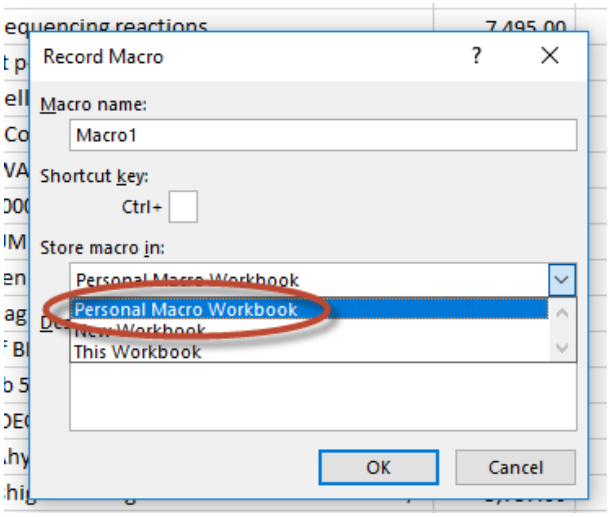
Setting up a Personal workbook

Having a personal workbook (which is saved on your profile) means that you can use Macros you have created across different workbooks.

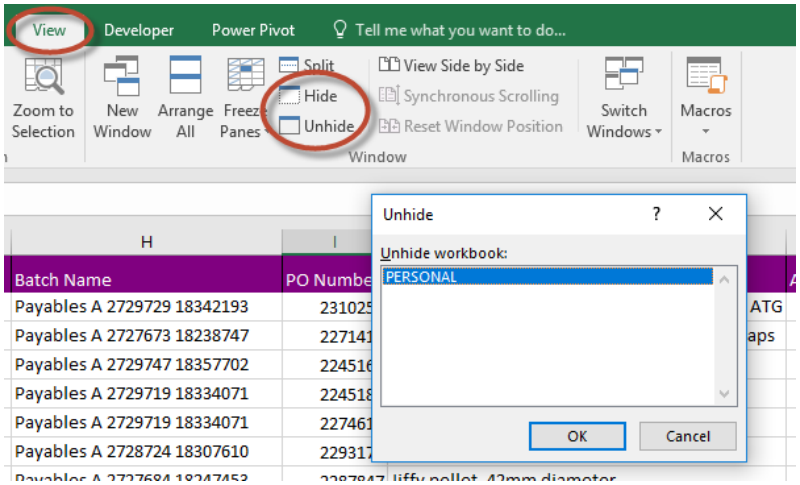
If you have never used the Personal Workbook before you will need to set it up

Open the **Developer tab** and click **Record Macro**





Select the Personal Macro Workbook option



The Personal Workbook will open up behind any new Excel workbook, if it gets annoying you can hide it (the Macros will still run but you won't be able to edit them unless it is unhidden)

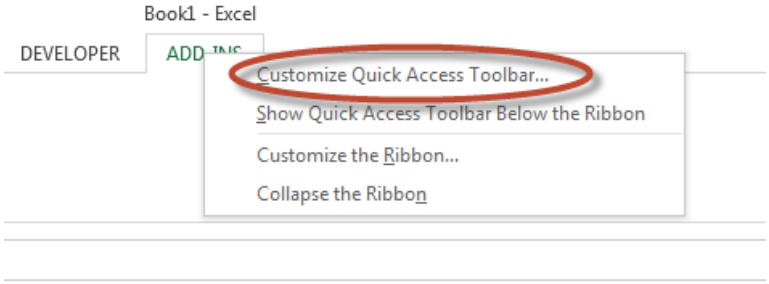
Warning

It is worth keeping a text copy (in a Word document) of any vital Macros. If your computer is upgraded or there is an issue with it, the Personal Macro Workbook could be lost. You would then need to recreate the Macros from the text file (which is much quicker than rewriting the Macro)

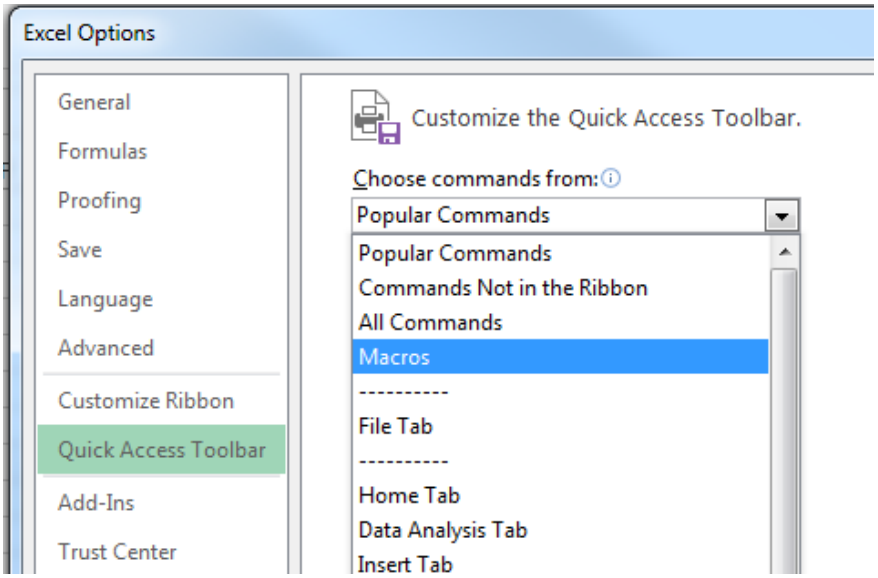
Adding a button to run the Macro

Add a button in the worksheet to run the Macro

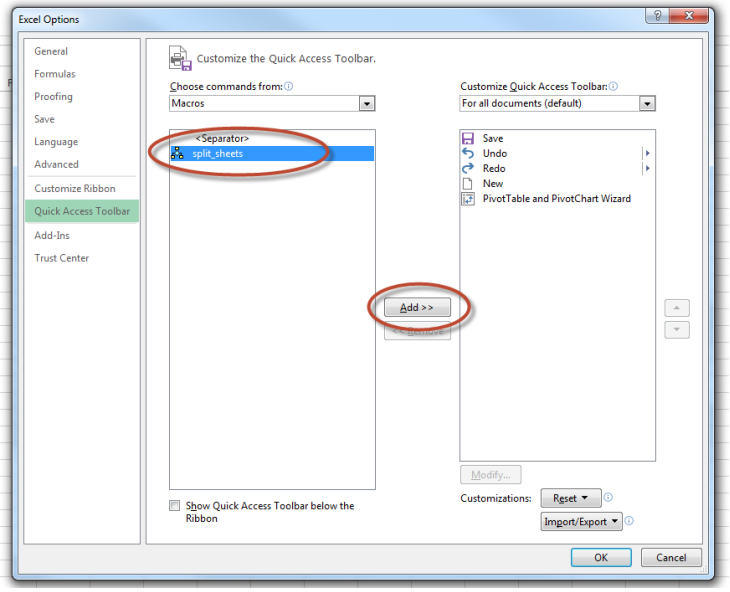
a) Right click on the Excel ribbon and select ***Customize Quick Access Toolbar***



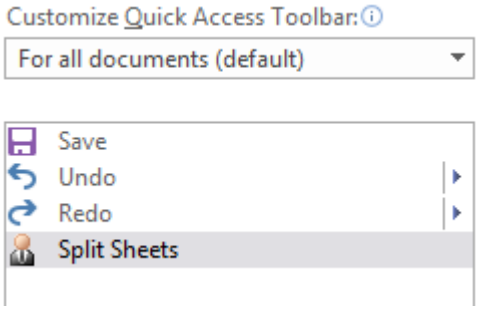
b) Select ***Macros*** from the choose commands from list



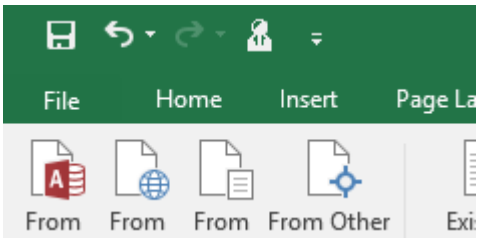
c) Find the macro required and **Add** it to your customized toolbar



d) Modify , select a suitable image and click **OK .**



e) It will now appear at the top of the screen



Create a macro to format numbers (an extra exercise if required)

In Parts 1 and 2 we often reformatted numbers to show negatives in red and in brackets. This is a time consuming process as the custom formats need to be altered to get the right format. It is possible to create a macro which will do the formatting automatically.

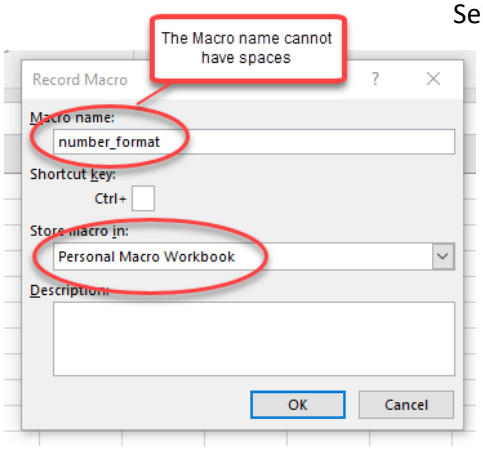
NB it is best to save this macro in the Personal Macro workbook so that it can be used on every workbook.

Step 1

	A	B	C	D	E	F	G	H	I
1	1046	33240	-727	22684	879	34777	12112	9732	22980
2	291	15936	25504	8622	24345	12098	10961	37965	14516
3	7539	11154	-609	25818	15329	25786	33287	40122	30524
4	5900	3671	31747	29693	43749	15158	43010	19957	30927
5	1804	15796	24722	8864	17071	28302	5819	2647	-2118
6	35332	37930	28301	29073	11387	1792	25426	39538	-5901
7	25234	8135	-9252	7561	7273	9775	-625	-7701	-8237
8	41063	29869	16600	9106	5566	34844	37685	36470	-4380
9	5550	4141	41539	-8913	36885	5984	718	43521	13434
10	-3470	-1292	28804	11858	40248	29091	33217	31625	39240
11	19765	-2009	36409	19494	4121	5441	13363	31558	12414
12	20092	42250	21792	24075	-3500	23705	3781	26386	27694
13	18423	25428	-1442	4186	-2437	-8276	28642	24905	-5372
14	30841	41239	44744	761	-4934	17730	2368	16079	-4384
15	9480	29560	32794	11575	14125	376	30386	18185	25056
16	259	24698	6710	10283	33888	23370	11381	39352	42029
17	14002	43903	29780	27687	15885	5057	12932	23177	41503
18	-8012	-7105	21669	42897	22730	12844	-836	31650	29746
19	13673	10333	26058	40159	31342	-8673	39960	-8953	20126
20	14741	11256	5431	-3367	5670	17145	282	23791	26992
21	7608	23128	23463	40540	32731	15518	6770	11640	-5918

Open an Excel workbook and add some random numbers (make sure there are some negatives so that the macro can be tested later).

Step 2

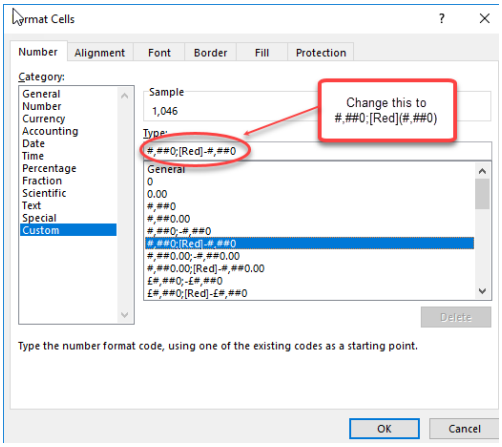


Select a single cell and start recording a macro (call it something you will remember later)

As mentioned above, it makes sense to store it in your Personal Macro Workbook

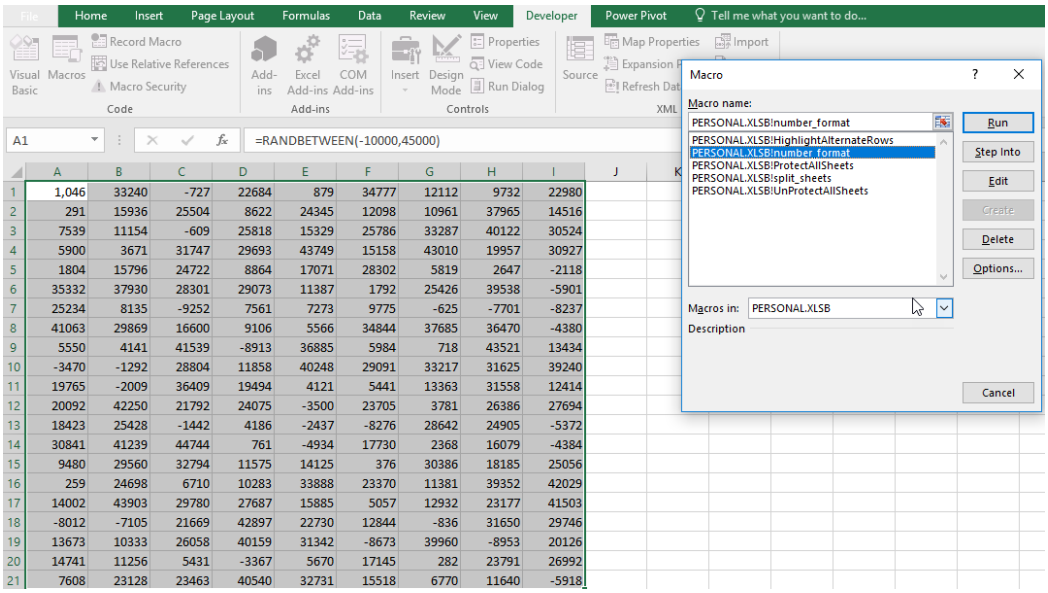
Step 3

While recording, format this first cell as required



Step 4

Select the data in the test workbook and run the macro to test it



Step 5

Add a button to the toolbar using the previous instructions

Exercise 4 – Creating a macro using Visual Basic code

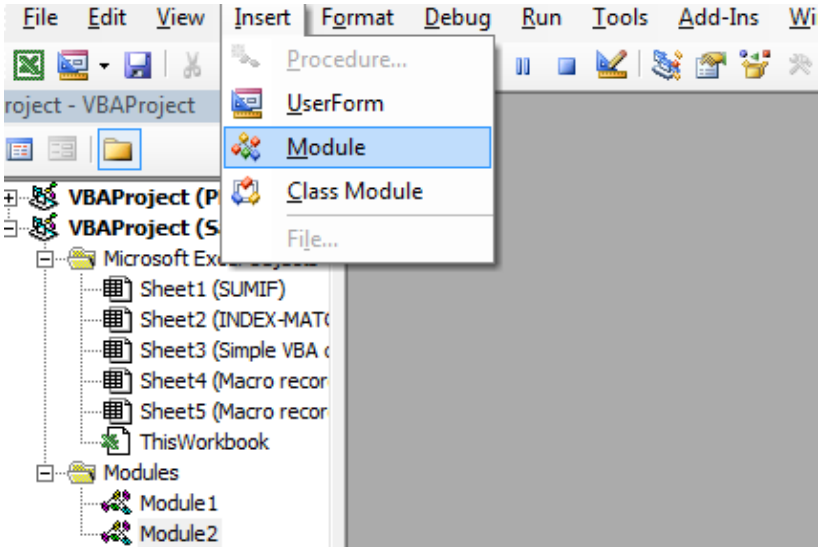
Rather than record your steps you can create (or edit) a macro using code known as 'Visual Basic'.

Task 1: Create a macro to protect all sheets

a) Open the workbook entitled **Ex4 source data**

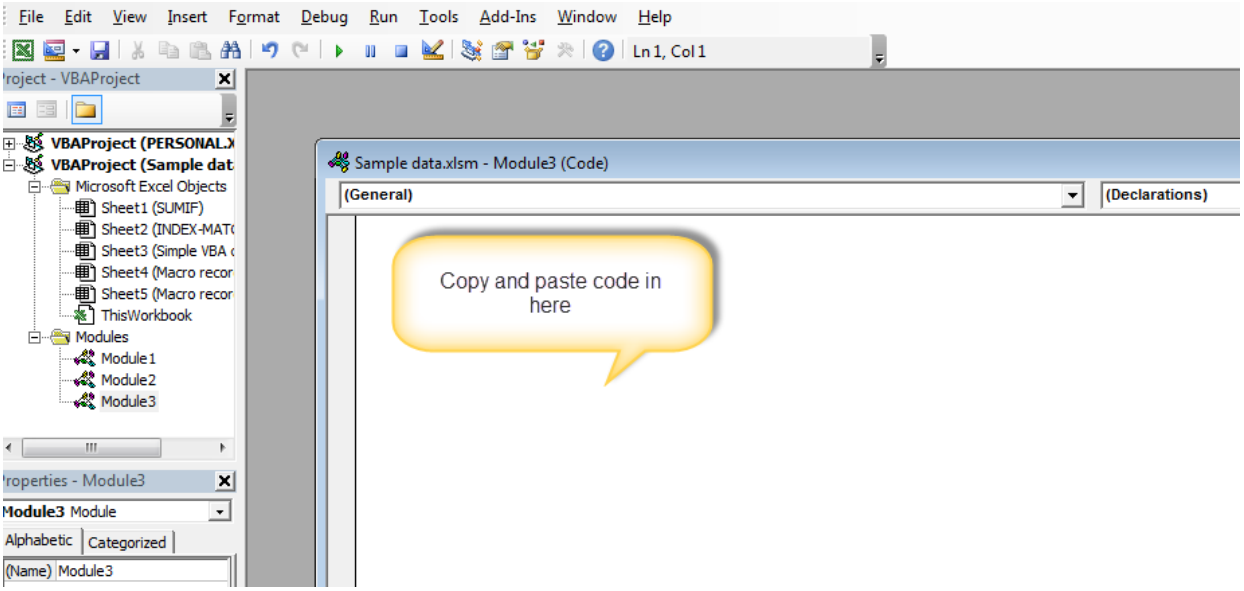
b) From the **Developer** tab select the **Visual Basic** button

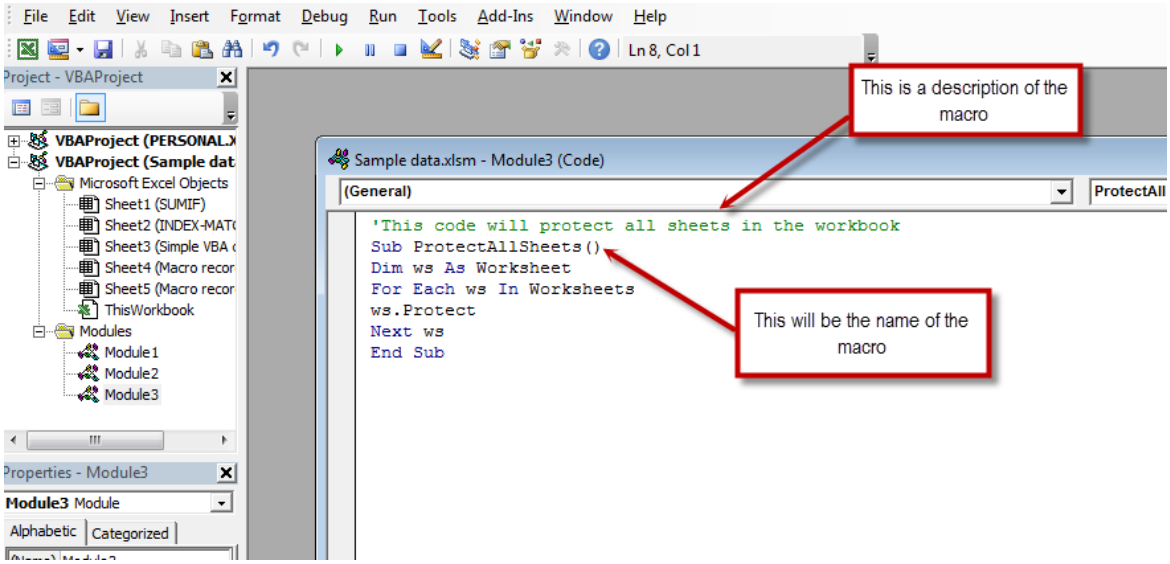
c) **Insert** the **Module** option



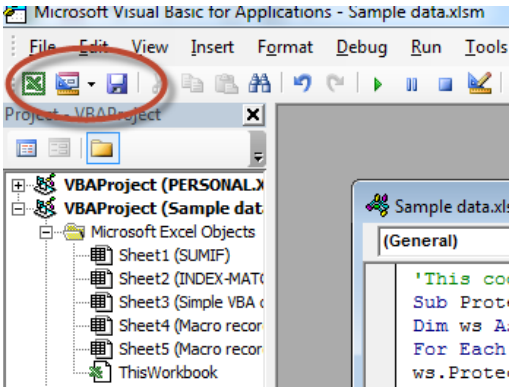
d)

e) Copy the **code** for ***Protect all worksheets*** (on page 17) into the visual editor box.



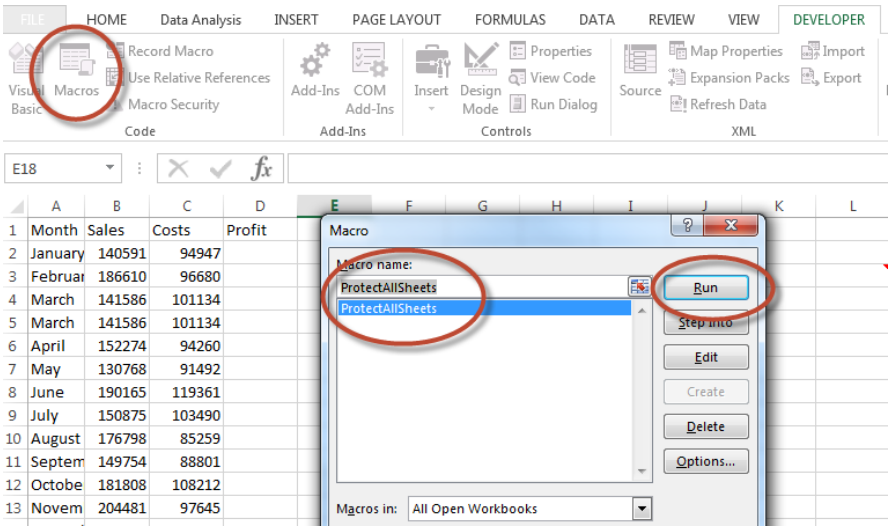


f) Save



g) Click on the **Excel** button to go back to your source data workbook

h) This macro is now available for you to run : simply click on the **Macros** button on the ribbon and select **Run**



Protect all worksheets

Use the below code to protect all the worksheets in a workbook at one go.

```
'This code will protect all sheets in the workbook
Sub ProtectAllSheets()
Dim ws As Worksheet
For Each ws In Worksheets
ws.Protect
Next ws
End Sub
```

This code will go through all the worksheets one by one and protect it.

To unprotect all the worksheets

Make sure you create a separate module and use ws.Unprotect instead of ws.Protect in the code and remember to update the macro description and title.

Highlighting alternates

This can increase the readability of your data, it is useful when you need to take a print out and review the data.

Here is a code that will instantly highlight alternate rows in the selection.

```
'This code would highlight alternate rows in the selection
Sub HighlightAlternateRows()
Dim Myrange As Range
Dim Myrow As Range
Set Myrange = Selection
For Each Myrow In Myrange.Rows
  If Myrow.Row Mod 2 = 1 Then
    Myrow.Interior.Color = vbCyan
  End If
Next Myrow
End Sub
```

Note that the specified colour is vbCyan in the code. You can specify other colors as well (such as vbRed, vbGreen, vbBlue).

Spell check

Excel doesn't have a spell check like Word or PowerPoint. While you can run the spell check by hitting the F7 key, there is no visual cue when there is a spelling mistake.

Use this code to instantly highlight all the cells that have a spelling mistake in it.

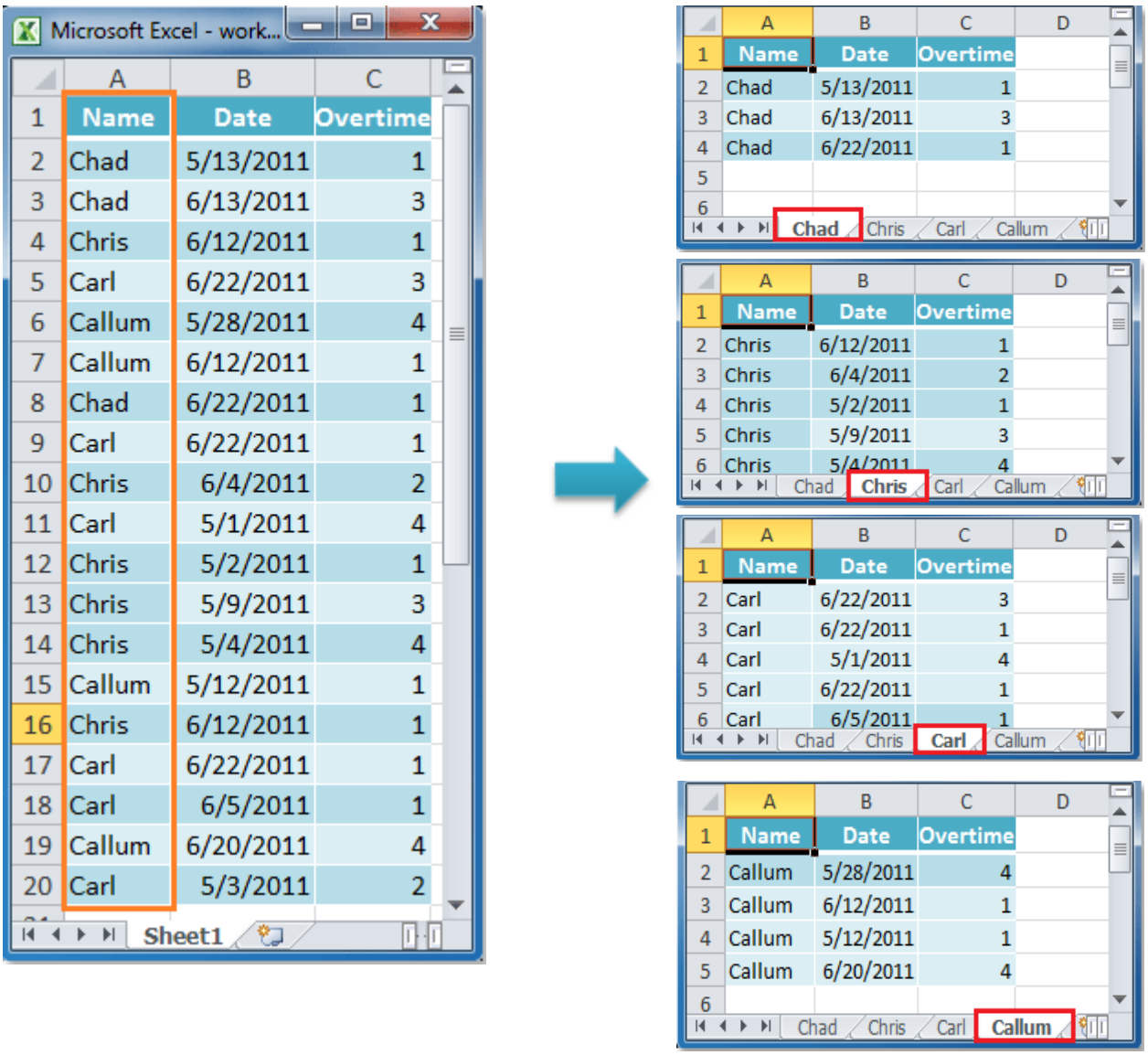
```
'This code will highlight the cells that have misspelled words
Sub HighlightMisspelledCells()
Dim cl As Range
For Each cl In ActiveSheet.UsedRange
If Not Application.CheckSpelling(word:=cl.Text) Then
cl.Interior.Color = vbRed
End If
Next cl
End Sub
```

Note that the cells that are highlighted are those that have text that Excel considers as a spelling error. In many cases, it would also highlight names or brand terms that it doesn't understand.

Exercise 5: Macros to split data onto multiple worksheets

Scenario

You have a worksheet with huge rows of data, and now, you need to split the data into multiple worksheets based on the **Name** column (see following screenshots), and the names are entered randomly. You could sort them first, and then copy and paste them one by one into other new worksheets.

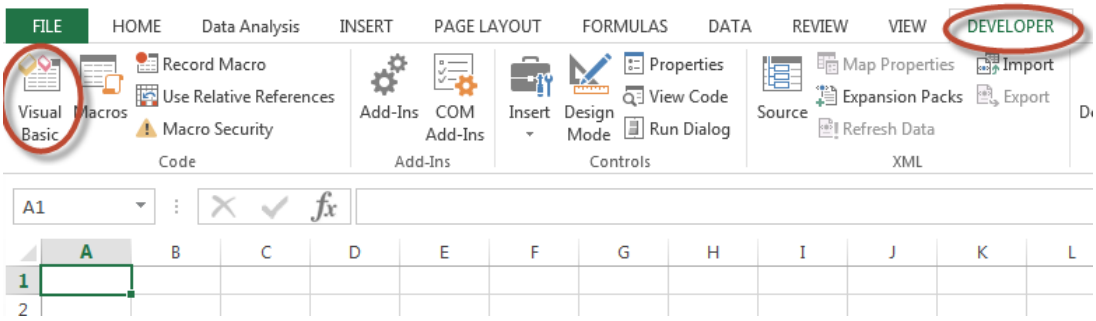


Name	Date	Overtime
Chad	5/13/2011	1
Chad	6/13/2011	3
Chris	6/12/2011	1
Carl	6/22/2011	3
Callum	5/28/2011	4
Callum	6/12/2011	1
Chad	6/22/2011	1
Carl	6/22/2011	1
Chris	6/4/2011	2
Carl	5/1/2011	4
Chris	5/2/2011	1
Chris	5/9/2011	3
Chris	5/4/2011	4
Callum	5/12/2011	1
Chris	6/12/2011	1
Carl	6/22/2011	1
Carl	6/5/2011	1
Callum	6/20/2011	4
Carl	5/3/2011	2

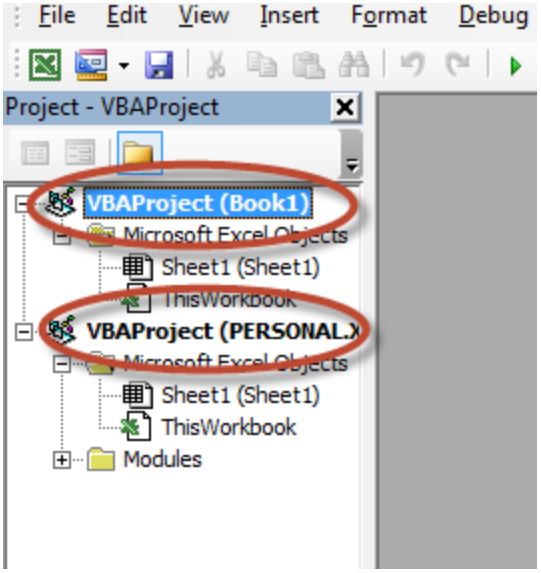
Task 1: Macro to split a workbook into separate worksheets

If you want to split the data based on column value quickly and automatically, the following VBA code can be used (the code was just found on the internet using a simple search).

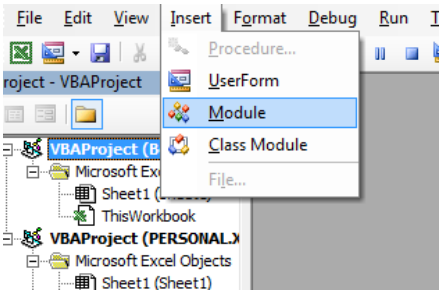
a) Save the worksheet as a macro enabled workbook.
b) Open the ***Microsoft Visual Basic for Applications*** window.



c) Decide if you want to save the macro in the current workbook (just available while this workbook is open, or to a Personal Macro Workbook, which will be available in any Workbook. (see page Setting up a Personal workbook)



d) Click **Insert** > **Module**, and paste the following code in the Module Window.



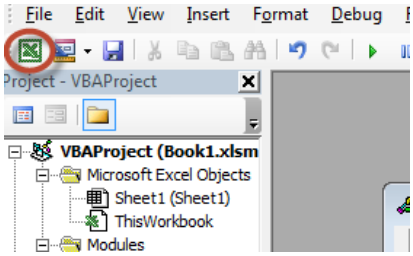
```
Sub split_sheets()
Dim lr As Long
Dim ws As Worksheet
Dim vcol, i As Integer
Dim icol As Long
Dim myarr As Variant
Dim title As String
Dim titlerow As Integer
vcol = 1
Set ws = Sheets("Sheet1")
lr = ws.Cells(ws.Rows.Count, vcol).End(xlUp).Row
title = "A1:C1"
titlerow = ws.Range(title).Cells(1).Row
icol = ws.Columns.Count
ws.Cells(1, icol) = "Unique"
For i = 2 To lr
On Error Resume Next
If ws.Cells(i, vcol) <> "" And Application.WorksheetFunction.Match(ws.Cells(i, vcol),
ws.Columns(icol), 0) = 0 Then
ws.Cells(ws.Rows.Count, icol).End(xlUp).Offset(1) = ws.Cells(i, vcol)
End If
Next
myarr =
Application.WorksheetFunction.Transpose(ws.Columns(icol).SpecialCells(xlCellTypeConstants))
ws.Columns(icol).Clear
For i = 2 To UBound(myarr)
ws.Range(title).AutoFilter field:=vcol, Criteria1:=myarr(i) & ""
If Not Evaluate("=ISREF('" & myarr(i) & "'!A1)") Then
Sheets.Add(after:=Worksheets(Worksheets.Count)).Name = myarr(i) & ""
Else
Sheets(myarr(i) & "").Move after:=Worksheets(Worksheets.Count)
End If
ws.Range("A" & titlerow & ":A" & lr).EntireRow.Copy Sheets(myarr(i) & "").Range("A1")
Sheets(myarr(i) & "").Columns.AutoFit
Next
ws.AutoFilterMode = False
ws.Activate
End Sub
```

Note: In the above code:

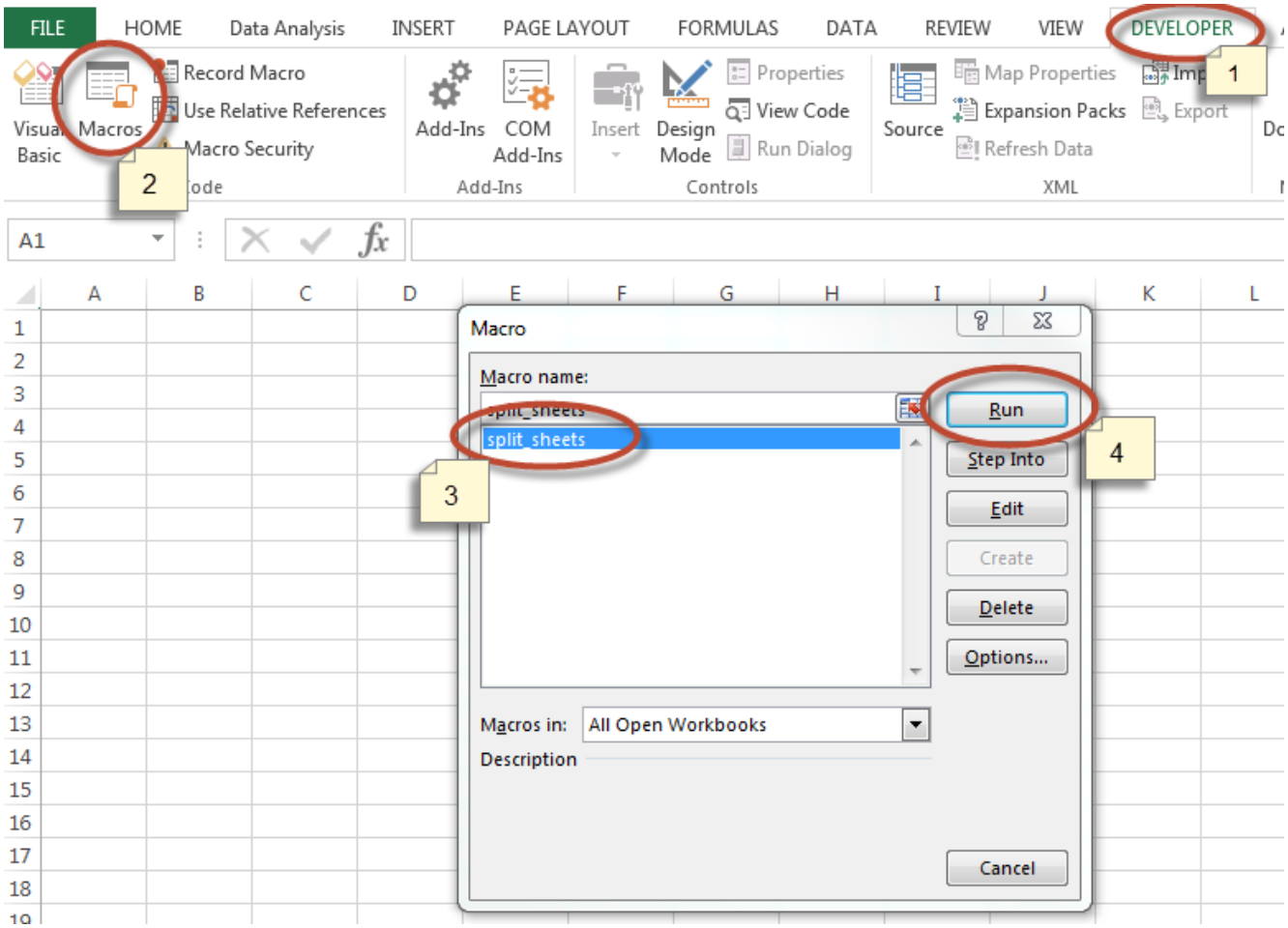
a) vcol =1, the number **1** is the column number that contains the data that you want to use as the basis of your split
b) Set ws = Sheets("Sheet1"), **Sheet1** is the sheet name that you want to apply this code to
c) title = "A1:C1", **A1:C1** is the range of the title.

All of them are variables, you can change them as needed.

e) **Save** and return to the spreadsheet



f) Find and run the macro as before



Note: The split worksheets are placed in the end of the workbook where the master worksheet is.

Task 2: Split the sheets into different workbooks

a) Create and save a new macro using the VBA code below

```
Sub Splitbook()
Dim xPath As String
xPath = Application.ActiveWorkbook.Path
Application.ScreenUpdating = False
Application.DisplayAlerts = False
For Each xWs In ThisWorkbook.Sheets
   xWs.Copy
   Application.ActiveWorkbook.SaveAs Filename:=xPath & "\" & xWs.Name & ".xlsx"
   Application.ActiveWorkbook.Close False
Next
Application.DisplayAlerts = True
Application.ScreenUpdating = True
End Sub
```

b) Open the ***Ex5 source data*** workbook

c) Create a macro which uses the above code to split the workbook into different workbooks

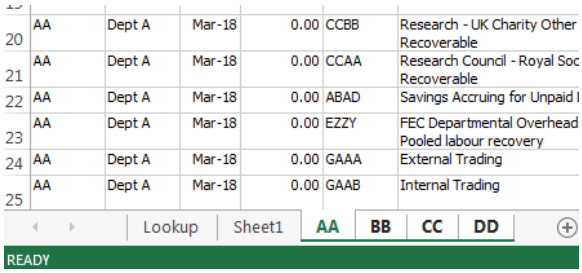
(NB this macro needs to be created in the workbook where it will be used)

Exercise 6: Consolidation

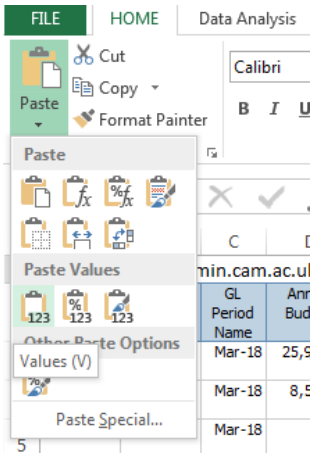
This exercise follows on from Exercise 5 (a correct version of the split sheet workbook is available if needed)

Task 1: Recorder tool macro

1. Copy the data from the separate ***email look up data*** file into a new sheet in the workbook (call this sheet Lookup).
2. Select all the department sheets



3. Use the Macro recorder tool to write a macro which:
    - Inserts a new row at the top of each sheet
    - Adds the relevant email address from the lookup table
    - Use copy paste values to ensure that the email and not the formula is in A1



4. Delete the new rows and test that the macro works as expected

Task 2: VBA macro

Use VBA to create a macro which will email the individual sheets to the email addresses now input in A1. (NB the code has been downloaded from an internet search)

When creating the macro in VBA, change the subject to 'Excel Part 3_Yourname'

```
Sub Mail_Every_Worksheet()
'Working in Excel 2000-2016
'For Tips see: http://www.rondebruin.nl/win/winmail/Outlook/tips.htm
   Dim sh As Worksheet
   Dim wb As Workbook
   Dim FileExtStr As String
   Dim FileFormatNum As Long
   Dim TempFilePath As String
   Dim TempFileName As String
   Dim OutApp As Object
   Dim OutMail As Object

   TempFilePath = Environ$("temp") & "\"

   If Val(Application.Version) < 12 Then
      'You use Excel 97-2003
      FileExtStr = ".xls": FileFormatNum = -4143
   Else
      'You use Excel 2007-2016
      FileExtStr = ".xlsm": FileFormatNum = 52
   End If

   With Application
      .ScreenUpdating = False
      .EnableEvents = False
   End With

   Set OutApp = CreateObject("Outlook.Application")

   For Each sh In ThisWorkbook.Worksheets
      If sh.Range("A1").Value Like "?*@?*.?*" Then

         sh.Copy
         Set wb = ActiveWorkbook

         TempFileName = "Sheet " & sh.Name & " of " _
                  & ThisWorkbook.Name & " " & Format(Now, "dd-mmm-yy h-mm-ss")

         Set OutMail = OutApp.CreateItem(0)

         With wb
            .SaveAs TempFilePath & TempFileName & FileExtStr, FileFormat:=FileFormatNum

            On Error Resume Next
            With OutMail
               .to = sh.Range("A1").Value
               .CC = ""
               .BCC = ""
               .Subject = "This is the Subject line"
               .Body = "Hi there"
               .Attachments.Add wb.FullName
```

```
            'You can add other files also like this
            '.Attachments.Add ("C:\test.txt")
            .Send   'or use .Display
        End With
        On Error GoTo 0

        .Close savechanges:=False
    End With

    Set OutMail = Nothing

    Kill TempFilePath & TempFileName & FileExtStr

  End If
Next sh

Set OutApp = Nothing

With Application
  .ScreenUpdating = True
  .EnableEvents = True
End With
End Sub
```

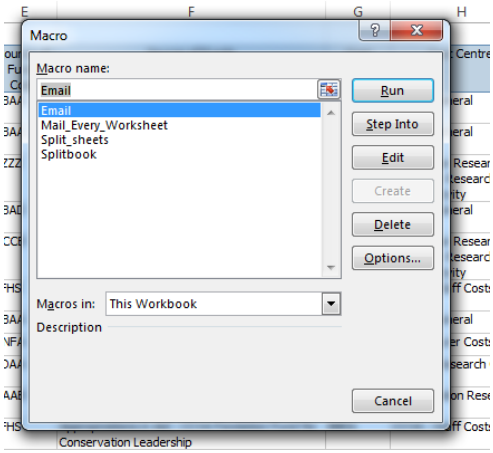
Exercise 7: Master macro

Create a master macro that runs:

- The macro to split the workbook into sheets
- The macro to add the email address to the top of each sheet

Step 1

Copy all of the necessary Macros into the specific workbook (some may previously have been in Personal or other workbooks) so they are all in one place



Step 2

Launch the Visual Basic editor

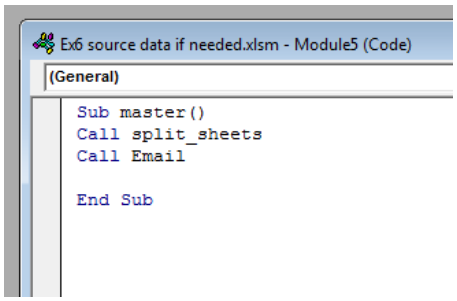
Step 3

The first line is **Sub Master()**

Under this line, type "Call," followed by the name of the first macro you would like to run e.g. **Call Split_sheets()**

Press "Enter" to go to the next line. Type "Call," followed by the name of the second macro e.g. **Call Email()**

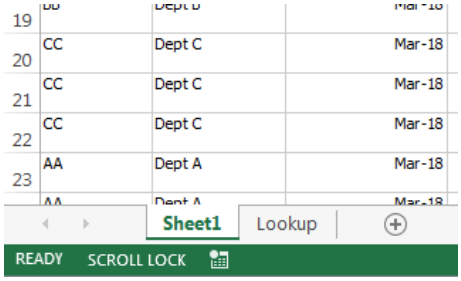
Continue until all of the required macros are included



Save the macro and close the Visual Basic editor.

Step 4

Delete the separate department sheets



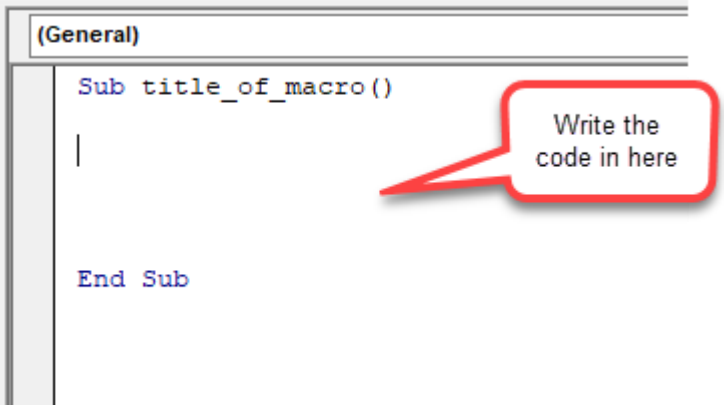
Step 5

Run the master macro to rest that it works

Appendix 1: Writing macros

Setting up the macro

- Start the macro with **Sub name()**
- End the macro with **End Sub**
- Write the code in the middle



Move to different cells

In a macro it is often useful to move to different cells to either run the same action or to select it. The ActiveCell or ActiveSheet commands can be used.

ActiveSheet.range("G2").Select would move the cursor to cell G2

ActiveCell.Offset(1,0).Select would select the cell in the row below (the first number relates to the row and second number relates to the column)

ActiveCell.Offset(0,1).Select would select the cell in the next column to the right (on the same row)

ActiveCell.Offset(-1,0).Select would select the cell in the row above

Looping

A very useful action is to loop through a range performing a certain task until a specific condition exists. i.e. perform the same action in each cell until a blank cell is reached.

Start the code with **Do Until** [add the condition in here]

Finish with **Loop**

Write the code for the action in the middle

Example:

Make the cell content bold, move to the next cell down, repeat the action. Continue until the next cell is empty

```
Sub bold()
Do Until ActiveCell.Offset(1, 0).Value = ""
Selection.Font.bold = True
ActiveCell.Offset(1, 0).Select
Selection.Font.bold = True
Loop
End Sub
```

The macro is called bold

Continue the loop until the cell below is blank

Make the font in the selected cell, bold

Select the next cell in the column and make the font bold

Loop and repeat until the **Do Until** condition is met (in this case, the next cell is blank)

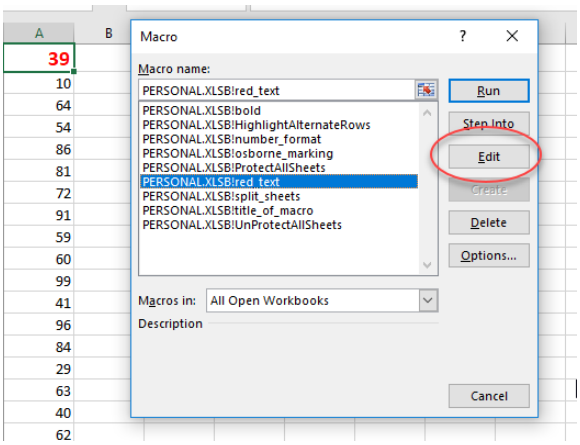
Tip for finding the code for specific formatting

If you want to format cells in a specific way but don't know the code, use the macro recorder function.

- Start recording a macro (use a name you will find later)
- Format the cell in the required way
- Stop recording
- Find the macro and copy the code into your own macro

E.g you want to format cells with red bold text, size 14

1. Set up a spreadsheet with some test data
2. Start recording a macro (call it red_text)
3. Format one cell in the required format
4. Stop recording
5. Find the macro and copy the code to use in your macro




```
Sub red_text()
'
' red_text Macro
'
'
    With Selection.Font
        .Color = -16776961
        .TintAndShade = 0
    End With
    Selection.Font.bold = True
    Selection.Font.Size = 14

End Sub
```

```
(General)
Sub red_text()

Do Until ActiveCell.Offset(1, 0).Value = ""

    With Selection.Font
        .Color = -16776961
        .TintAndShade = 0
    End With
    Selection.Font.bold = True
    Selection.Font.Size = 14

ActiveCell.Offset(1, 0).Select

    With Selection.Font
        .Color = -16776961
        .TintAndShade = 0
    End With
    Selection.Font.bold = True
    Selection.Font.Size = 14

Loop

End Sub
```

This macro will make the text bold red, size 14 in every non blank cell in the column.