Continue

Combined Data is Better DataStop me if youve heard this one before: I need to make a pivot table but the data is spread out in lots of different Excel filesDamnit!This situation blows, since your analysis depends on a pivot table and combining a bunch of workbooks together by hand is terrible.Fortunately, VBA will make short work of this pain in the ass. Here goes:Heres a link to the code above so you can review it side-by-side with the walk through below. Right-click, Open in new window.Lets break this challenge down step-by-step using the 4-step VBA process as our guide:Step 1 SetupStep 2 ExplorationStep 3 ExecutionStep 4 CleanupOur Step 1 Setup is covered by lines 16-18 short and sweet.First, on line 16, we assign the folder name where the individual Excel files are stored. (Youll want to change this to your folder, but in this example we are targeting C:\blog\example_data_in_here.)Then, on lines 16-17, we create a new Workbook (where Dst is short for destination, i.e. output) to store the data from each individual file, then assign the first Worksheet in that Workbook as the Dst Worksheet.Boom! One down, three to go.Step 2 Exploration begins on line 21, where we take advantage of the Dir function to loop through the directory we set up moments ago (StrDirContainingFiles) and identify every file that ends in .xlsx. (Thats what the asterisk character, *, is doing there at the end of the line.)Lines 22 through 25 store each file name inside a Collection (named colFileNames), which will make it SUPER easy to iterate through each file a little bit later in the code.Umm why not do the whole thing in this Dir loop?Yes, we COULD have conducted the bulk of the code inside this Dir loop, but I prefer using a Collection here because it reduces the number of nested loops in our subroutine. Each nested loop you add is another layer of complexity for you to mentally keep track of fuck that. Programming is hard enough avoid deeply-nested loops whenever you can.The use a Collection strategy also makes it really easy to verify that the loop worked and pulled in the data we expected, which you can check for yourself by un-commenting lines 27-31.Lets keep it moving though With those file names stored neatly in colFileNames, we begin looping through it on line 35.On line 38, strFilePath is assigned to be the original source folder string (strDirContainingFiles, which is C:\blog\example_data_in_here in this example), a backslash (\), and the file name from colFileNames.Well immediately take advantage of that full file path to the Excel file on line 41, where we open that Workbook and save a reference to it as wbkSrc (where Src is short for Source). Line 42 assigns the target worksheet, named data in this example, to wksSrc.Exploration continues on lines 46-47, where we take advantage of the LastOccupiedRowNum and LastOccupiedColNum functions (which are defined at the very bottom as well as in the VBA Toolbelt, which you should be using) to easily identify the last-occupied row and last-occupied column on the source Worksheet.This is critical here! By dynamically determining the last column and last row on each loop, we can be confident that were getting all the data from each Worksheet.With the last-occupied row and last-occupied column numbers stored in lngSrcLastRow and lngSrcLastCol respectively, we store the full data range on lines 48-51, starting from the top-left corner and extending to the bottom-right.This is another great checkpoint: we can verify that all the data has been correctly identified and stored in rngSrc by un-commenting out lines 53-55 and calling rngSrc.Select to highlight all the cells. Smooth!Heres where things get a bit more interestingOn lines 60-62, we check to see if this is NOT the first iteration.Why are we checking to see if this is the first loop? Why? Good question!On the first loop, we want to include the headers, but each subsequent time we do NOT want to include the headers.Each loop after the first, we adjust rngSrc to skip the first row like this: Set rngSrc = rngSrc.Offset(1, 0).Resize(rngSrc.Rows.Count 1)Heres the two-step process in slow-motion:.Offset(1, 0): this shifts the whole range down one row, meaning the first row is no longer included, but also means we now have a blank row at the bottom of rngSrc.Resize(rngSrc.Rows.Count 1): this adjusts the bottom row of rngSrc up one row by reducing the total count of rows that are included in the RangeNice!This puts us at another good checkpoint, where we can verify that the header is no longer part of rngSrc by un-commenting out lines 64-68.Phew with that, Step 2 Exploration is complete!Lets dive into Step 3 Execution, which kicks off on line 73.Here, we again check the iteration. If this is the first loop, our target cell is easy its A1, since the Worksheet is empty. (Thats what we handle on lines 74 and 75.) On the other hand, if this is NOT the first loop, then we:Again take advantage of LastOccupiedRowNum, which comes with the VBA Toolbelt and works on ANY Worksheet, to identify the last-occupied (on line 77)Set the target Range to be one cell down from the last-occupied row (on line 78)The actual copy / paste step happens on line 80, where we call rngSrc.Copy and pass in rngDst (which we just set on line 78) as the Destination.Almost done with this one, I promise!Almost there you guys, stay with me!The last challenge within Step 3 Execution is to add a column identifying which data file a given row came from by writing the Worksheet name into a far-right column. Lets get to it!Lines 86-89 cover another first loop special case if this is the first time through, then we need to make sure we add a header name! By taking advantage of LastOccupiedColNum, which, again, is implemented for you both below AND in the VBA Toolbelt, we know that lngDstLastCol + 1 gives us the column right next to the last-occupied column.We name this column header Source Filename on line 88.Since we know that each row of data from the last paste (which happened on line 80) came from one of the different Excel files, we can take advantage of the Range.Value property to quickly write the file name to each of those rows.First, we need to identify the first row of data that was just pasted in.We used this same exact row number back on line 78, so we essentially copy that logic and assign lngDstFirstFileRow to be lngDstLastRow + 1.Next, we need to figure out the last row that will get this file name data, and on line 103 we do just that.Guess which function rides in to our rescue? Yep its our old standby LastOccupiedRowNum.Now that we know the first row and the last row of the range of cells that will need to be populated with the file name, all thats left to do is get the right column number!And wouldnt you know it, LastOccupiedColNum, on line 104, assigns lngDstLastCol that exact value Now that we know the three critical components for a Range:The first row, which is stored in lngDstFirstFileRowThe last row, which is stored in lngDstLastRowThe column those rows need to be applied in, which is stored in lngDstLastCol We can write the file name easy peasy!On lines 107-108, we use the values from #1, #2, and #3 above to store the target Range.Damn son! That brings us to a great checkpoint by un-commenting lines 112-113, we can easily verify (using Range.Select) that the correct Range is defined.Finally, the last Execution task occurs on line 117, where we actually do the file name writing. Calling wbkSrc.Name returns the file name (in the first case, it will be AT_Apr_16.xlsx), which is why we assign the rngFile.Value to it.Jackpot Step 3 Execution is all done!And of course, we wrap up with Step 4 Cleanup.Nice and short here too: on line 122, we close the source data Workbook (with SaveChanges set to False, since we do not want to modify those files at all), and on line 127 we throw a quick Data combined! message box to let the user know the job is done!Want to see this code in action? Heres an 14-minute video guide:Combining many individual Excel files into a single file with VBA smoothly? If not, let me know and Ill help you get what you need! And if youd like more step-by-step, no-bullshit VBA guides delivered direct to your inbox, join my email newsletter below. This tutorial will show you how to combine multiple Excel files into one workbook in VBA.Creating a single workbook from a number of workbooks, using VBA requires a number of steps to be followed.You need to select the workbooks from which you want the source data the Source files.You need to select or create the workbook to which you wish to put the data the Destination file.You need to select the sheets from the Source files that you require.You need to tell the code where to place the data in the Destination file.Combining all Sheets from all Open Workbooks to a New Workbook as Individual SheetsIn the code below, the files you need to copy the information from need to be open as Excel will loop through the open files and copy the information into a new workbook. The code is placed in the Personal Macro Workbook.These files are the ONLY Excel Files that should be open.Sub CombineMultipleFiles()On Error GoTo eh'declare variables to hold the objects required Dim wbDestination As Workbook Dim wbSource As Workbook Dim wsSource As Worksheet Dim wb As Workbook Dim sh As Worksheet Dim strSheetName As String Dim strDestName As String'turn off the screen updating to speed things up Application.ScreenUpdating = False'first create new destination workbook Set wbDestination = Workbooks.Add'get the name of the new workbook so you exclude it from the loop below strDestName = wbDestination.Name'now loop through each of the workbooks open to get the data but exclude your new book or the Personal macro workbook For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then Set wbSource = wb For Each sh In wbSource.Worksheets sh.Copy After:=Workbooks(strDestName).Sheets(1) Next sh End If Next wb'now close all the open files except the one you want For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then wb.Close False End If Next wb'remove sheet one from the destination workbook Application.DisplayAlerts = False Sheets("Sheet1").Delete Application.DisplayAlerts = True'clean up the objects to release the memory Set wbDestination = Nothing Set wbSource = Nothing Set wsSource = Nothing Set wb = Nothing'turn on the screen updating when complete Application.ScreenUpdating = FalseExit Sub'Click on the Macro dialog box to run the procedure from your Excel screen.Your combined file will now be displayed.This code has looped through each file, and copied the sheet to a new file. If any of your files have more than one sheet it will copy those as well including the sheets with nothing on them!Combining all Sheets from all Open Workbooks to a Single Worksheet in a New WorkbookThe procedure below combines the information from all the sheets in all open workbooks into a single worksheet in a new workbook that is created.The information from each sheet is pasted into the destination sheet at the last occupied row on the worksheet.Sub CombineMultipleSheets()On Error GoTo eh'declare variables to hold the objects required Dim wbDestination As Workbook Dim wbSource As Workbook Dim wsDestination As Worksheet Dim wb As Workbook Dim sh As Worksheet Dim strSheetName As String Dim strDestName As String Dim iRws As Integer Dim iCols As Integer Dim totRws As Integer Dim strEndRng As String Dim rngSource As Range'turn off the screen updating to speed things up Application.ScreenUpdating = False'first create new destination workbook Set wbDestination = Workbooks.Add'get the name of the new workbook so you exclude it from the loop below strDestName = wbDestination.Name'now loop through each of the workbooks open to get the data For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then Set wbSource = wb For Each sh In wbSource.Worksheets'get the number of rows in the sheet sh.Activate ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Activate iRws = ActiveCell.Row iCols = ActiveCell.Column'get the range of the last cell in the sheet strEndRng = sh.Cells(iRws, iCols).Address'set the source range to copy Set rngSource = sh.Range("A1:" & strEndRng)'find the last row in the destination sheet wbDestination.Activate Set wsDestination = ActiveSheet wsDestination.Cells.SpecialCells(xlCellTypeLastCell).Select totRws = ActiveCell.Row'check if there are enough rows to paste the data If totRws + rngSource.Rows.Count > wsDestination.Rows.Count Then MsgBox "There are not enough rows to place the data in the Consolidation worksheet." GoTo eh End If'add a row to paste on the next row down If totRws 1 Then totRws = totRws + 1 rngSource.Copy Destination:=wsDestination.Range("A" & totRws) Next sh End If Next wb'now close all the open files except the one you want For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then wb.Close False End If Next wb'clean up the objects to release the memory Set wbDestination = Nothing Set wbSource = Nothing Set wsDestination = Nothing Set rngSource = Nothing Set wb = Nothing'turn on the screen updating when complete Application.ScreenUpdating = FalseExit Sub'Combining all Sheets from all Open Workbooks to a Single Worksheet in an Active WorkbookIf you want to bring the information from all other open Workbooks in to the one you are currently working in, you can use this code below.Sub CombineMultipleSheetsToExisting() On Error GoTo eh'declare variables to hold the objects required Dim wbDestination As Workbook Dim wbSource As Workbook Dim wsDestination As Worksheet Dim wb As Workbook Dim sh As Worksheet Dim strSheetName As String Dim strDestName As String Dim iRws As Integer Dim iCols As Integer Dim totRws As Integer Dim rngEnd As String Dim rngSource As Range'set the active workbook object for the destination book Set wbDestination = ActiveWorkbook'get the name of the active file strDestName = wbDestination.Name'turn off the screen updating to speed things up Application.ScreenUpdating = False'first create new destination worksheet in your Active workbook Application.DisplayAlerts = False'resume next error in case sheet doesn't exist On Error Resume Next ActiveWorkbook.Sheets("Consolidation").Delete'reset error trap to go to the error trap at the end On Error GoTo eh Application.DisplayAlerts = True'add a new sheet to the workbook With ActiveWorkbook Set wsDestination = .Sheets.Add(After:=.Sheets(.Sheets.Count)) wsDestination.Name = "Consolidation" End With'now loop through each of the workbooks open to get the data For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then Set wbSource = wb For Each sh In wbSource.Worksheets'get the number of rows in the sheet sh.Activate ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Activate iRws = ActiveCell.Row iCols = ActiveCell.Column rngEnd = sh.Cells(iRws, iCols).Address Set rngSource = sh.Range("A1:" & rngEnd)'find the last row in the destination sheet wbDestination.Activate Set wsDestination = ActiveSheet wsDestination.Cells.SpecialCells(xlCellTypeLastCell).Select totRws = ActiveCell.Row'check if there are enough rows to paste the data If totRws + rngSource.Rows.Count > wsDestination.Rows.Count Then MsgBox "There are not enough rows to place the data in the Consolidation worksheet." GoTo eh End If'add a row to paste on the next row down if you are not in row 1 If totRws 1 Then totRws = totRws + 1 rngSource.Copy Destination:=wsDestination.Range("A" & totRws) Next sh End If Next wb'now close all the open files except the one you want For Each wb In Application.Workbooks If wb.Name strDestName And wb.Name "PERSONAL.XLSB" Then wb.Close False End If Next wb'clean up the objects to release the memory Set wbDestination = Nothing Set wbSource = Nothing Set wsDestination = Nothing Set rngSource = Nothing Set wb = Nothing'turn on the screen updating when complete Application.ScreenUpdating = FalseExit Sub'eh: MsgBox Err.DescriptionEnd Sub Stop searching for VBA code online. Learn more about AutoMacro - A VBA Code Builder that allows beginners to code procedures from scratch with minimal coding knowledge and with many time-saving features for all users! Learn More! Below is the VBA code to merge multiple excel files which are entered in a folder in D Drive & the Folder name is Files. So if your folder name is different then you can change the path according to your drive. Input Data Final Output After Merge VBA CODE Sub MergeWorkbooks()Dim FolderPath As StringDim File As StringDim i As LongFolderPath = "D:\Files\"File = Dir(FolderPath)Do While File "" Workbooks.Open FolderPath & File ActiveWorkbook.Worksheets(1).Copy _ after:=ThisWorkbook.Worksheets(ThisWorkbook.Worksheets.Count) ActiveSheet.Name = Replace(File, ".xlsx", "") Workbooks(File).Close File = Dir()LoopEnd Sub This VBA code will help you to consolidate the All Excel Sheets data in One File, but the limitation is that if you excel data in one file but different-different sheets. then this code will consolidate all sheets data from one to a single sheet. Input Data Final Output After Merge VBA CODE Sub CombineSheets() Dim ws As Worksheet Dim combinedSheet As Worksheet Dim lastRow As Long, combinedLastRow As Long Dim sheetName As String ' Add a new sheet to store combined data Set combinedSheet = ThisWorkbook.Sheets.Add(After:= _ ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)) combinedSheet.Name = "MergedData" ' Loop through each sheet in the workbook For Each ws In ThisWorkbook.Worksheets ' Skip the combined sheet itself If ws.Name combinedSheet.Name Then ' Get the last row of data in the combined sheet combinedLastRow = combinedSheet.Cells(Rows.Count, "A").End(xlUp).Row ' Get the sheet name sheetName = ws.Name ' Loop through each row of data in the current sheet For lastRow = 1 To ws.Cells(Rows.Count, "A").End(xlUp).Row ' Copy data to the combined sheet combinedSheet.Cells(combinedLastRow + lastRow - 1, 1).Value = sheetName ws.Rows(lastRow).Copy Destination:=combinedSheet.Rows(combinedLastRow + lastRow - 1) Next lastRow End If Next ws MsgBox "All sheets combined successfully!", vbInformationEnd Sub In this VBA code i have provided a Folder picker, that means when you run this VBA code, then it will show a dialogue box that will ask for the folder path, then you have to select the folder from which you want to combine all the files in one file. after clicking the folder you just click on Ok Button then the process will start and all excel files will be merged into one excel file but in different-different sheets. Merge with Folder Picker Final Output After Merge VBA CODE Sub CombineWorkbooks() Dim myDialog As FileDialog, myFolder As String, myFile As String Set myDialog = Application.FileDialog(msoFileDialogFolderPicker) If myDialog.Show = -1 Then myFolder = myDialog.SelectedItems(1) & Application.PathSeparator myFile = Dir(myFolder & "*.xls*") Do While myFile "" Workbooks.Open myFolder & myFile Workbooks(myFile).Worksheets.Copy After:=ThisWorkbook.ActiveSheet Workbooks(myFile).Close myFile = Dir LoopEnd If End Sub This VBA code will help you to merge all Excel Files data from a specific folder, into one Excel file. In the below VBA code, i have given a folder path in D Drive, which contains a Folder with the name Files. So if your folder name is different then you can change the path according to your drive. Input Data Final Output After Merge VBA CODE Sub MergeDataFromFolder()Dim copiedsheetcount As LongDim rowcnt As LongDim merged As WorkbookDim wb As WorkbookDim ws As Worksheetfilefolder = "D:\Files\"Filename = Dir(filefolder & "*.xlsx")If Filename = vbNullString ThenMsgBox prompt:="No File", Buttons:=vbCritical, Title:="error"Exit SubEnd Ifcopiedsheetcount = 0rowcnt = 1Set merged = Workbooks.AddActiveSheet.Name = "Merged Data"Do While Filename vbNullStringcopiedsheetcount = copiehsheetcount + 1Set wb = Workbooks.Open(Filename:=filefolder & Filename, UpdateLinks:=False)Set ws = wb.Worksheets(1)With wsIf FilterMode Then .ShowAllDataIf copiedsheetcount > 1 Then .Rows(1).EntireRow.Delete shift:=xlUp.Range("a1").CurrentRegion.Copy Destination:=merged.Worksheets(1).Cells(rowcnt, 1)End Withwb.Close savechanges:=Falserowcnt = Application.WorksheetFunction.CountA(merged.Worksheets(1).Columns("A:A")) + 1Filename = DirLoopMsgBox prompt:="File Merged", Buttons:=vbInformation, Title:="Success"End Sub 5. Merge All Sheets from Multiple Files into a Single Sheet With the help of this VBA code, you can combine or merge multiple Excel files and their sheets of data into one Excel sheet with one click. for example, you have 12 months of salary file data and have 3 sheets in each Excel file, in the 1st sheet is Delhi Branch Salary Data, in the 2nd sheet Mumbai Branch Salary data is given and in the 3rd sheet, Kolkata Branch Salary data is given. Now i want to combine and merge all the branches data with 12 months, which means i want to append 36 sheets of data in one sheet then you should try this code to append all sheets of data into one Excel sheet. Input Data Final Output After Merge VBA CODE Sub MergeAllFilesAndSheets() Dim folderPath As String Dim mergedWorkbook As Workbook Dim sourceWorkbook As Workbook Dim ws As Worksheet Dim fileName As String Dim mergedSheet As Worksheet Dim lastRow As Long Dim dataRange As Range folderPath = "D:\Files\" Set mergedWorkbook = Workbooks.Add Set mergedSheet = mergedWorkbook.Sheets(1) ' Made by TechGuruPlus.com fileName = Dir(folderPath & "*.xlsx") Do While fileName "" Set sourceWorkbook = Workbooks.Open(folderPath & fileName) For Each ws In sourceWorkbook.Sheets lastRow = mergedSheet.Cells(mergedSheet.Rows.Count, 1).End(xlUp).Row + 1 mergedSheet.Cells(lastRow, 1).Resize(ws.UsedRange.Rows.Count, 1).Value = ws.Name & " (" & sourceWorkbook.Name & ")" ws.UsedRange.Copy Destination:=mergedSheet.Cells(lastRow, 2) Next ws sourceWorkbook.Close False fileName = Dir Loop mergedWorkbook.SaveAs folderPath & "MergedWorkbook_WithSheetAndFileName.xlsx" mergedWorkbook.Close MsgBox "All sheets from all files merged successfully", vbInformationEnd Sub Download Files Tutorial Video 6. Merge Specific Sheets from Multiple Files into a Single Sheet VBA CODE Sub MergeSpecificSheetsFromFiles() Dim folderPath As String Dim mergedWorkbook As Workbook Dim sourceWorkbook As Workbook Dim ws As Worksheet Dim fileName As String Dim mergedSheet As Worksheet Dim lastRow As Long Dim sheetName As String Dim sheetFound As Boolean Dim mergedFilePath As String ' Prompt user to enter the sheet name sheetName = InputBox("Enter the sheet name you want to merge:", "Sheet Name") If sheetName = "" Then MsgBox "No sheet name entered. Exiting...", vbExclamation Exit Sub End If folderPath = "D:\Files\" Set mergedWorkbook = Workbooks.Add Set mergedSheet = mergedWorkbook.Sheets(1) fileName = Dir(folderPath & "*.xls*") ' Handles both .xls and .xlsx extensions sheetFound = False Do While fileName "" Set sourceWorkbook = Workbooks.Open(folderPath & fileName) On Error Resume Next Set ws = sourceWorkbook.Sheets(sheetName) On Error GoTo 0 If Not ws Is Nothing Then sheetFound = True lastRow = mergedSheet.Cells(mergedSheet.Rows.Count, 1).End(xlUp).Row + 1 mergedSheet.Cells(lastRow, 1).Resize(ws.UsedRange.Rows.Count, 1).Value = sourceWorkbook.Name ws.UsedRange.Copy Destination:=mergedSheet.Cells(lastRow, 2) End If sourceWorkbook.Close False fileName = Dir Loop If Not sheetFound Then MsgBox "This sheet is not found in any file.", vbExclamation mergedWorkbook.Close False Exit Sub End If mergedFilePath = folderPath & "MergedWorkbook_" & sheetName & ".xlsx" mergedWorkbook.SaveAs mergedFilePath mergedWorkbook.Close ' Open the merged workbook Workbooks.Open mergedFilePath MsgBox "All '" & sheetName & "' sheets from all files merged successfully", vbInformationEnd Sub Download VBA Code 7. Merge Specific Sheets from Multiple Workbooks to Separate Tabs VBA CODE Sub MergeSpecificSheetsToSeparateTabs() Dim folderPath As String Dim sourceWorkbook As Workbook Dim mergedWorkbook As Workbook Dim ws As Worksheet Dim sheetName As String Dim fileName As String Dim newSheet As Worksheet Dim sheetFound As Boolean Dim mergedFilePath As String ' Prompt user to enter the sheet name sheetName = InputBox("Enter the sheet name you want to merge:", "Sheet Name") If sheetName = "" Then MsgBox "No sheet name entered. Exiting...", vbExclamation Exit Sub End If ' Set the folder path folderPath = "D:\Files\" ' You can modify this to your folder path ' Create a new workbook to hold the merged sheets Set mergedWorkbook = Workbooks.Add mergedWorkbook.Sheets(1).Name = "Temp" ' Temporary sheet to delete later fileName = Dir(folderPath & "*.xls*") ' Handles both .xls and .xlsx extensions sheetFound = False Do While fileName "" Set sourceWorkbook = Workbooks.Open(folderPath & fileName) On Error Resume Next Set ws = sourceWorkbook.Sheets(sheetName) On Error GoTo 0 If Not ws Is Nothing Then sheetFound = True ' Add a new sheet to the merged workbook Set newSheet = mergedWorkbook.Sheets.Add(After:=mergedWorkbook.Sheets(mergedWorkbook.Sheets.Count)) newSheet.Name = Left(sourceWorkbook.Name, 31) ' Sheet names are limited to 31 characters ' Copy the content of the specific sheet from the source workbook to the new sheet ws.UsedRange.Copy Destination:=newSheet.Cells(1, 1) End If sourceWorkbook.Close False fileName = Dir Loop ' Delete the temporary sheet if at least one sheet was found If sheetFound Then Application.DisplayAlerts = False mergedWorkbook.Sheets("Temp").Delete Application.DisplayAlerts = True Else MsgBox "This sheet is not found in any file.", vbExclamation mergedWorkbook.Close False Exit Sub End If ' Save the merged workbook mergedFilePath = folderPath & "MergedWorkbook_" & sheetName & ".xlsx" mergedWorkbook.SaveAs mergedFilePath mergedWorkbook.Close ' Open the merged workbook Workbooks.Open mergedFilePath MsgBox "All '" & sheetName & "' sheets from all files have been merged into separate tabs.", vbInformationEnd Sub Download VBA Code Steps to use VBA code to merge Excel sheets data To enter the above VBA code follow the steps as given below- Go to VBA Application by pressing the shortcut ALT + F11 or Right click on any sheet name and click on View Code. Now Insert a New Module (Go to Insert Menu and Click on Module) Now Copy the above VBA code and Paste in the Blank Module File. Press F5 Button to Run the Macro. All the File data have been merged into one file. if you are facing any problem to merge your data, please write us in the comment box below, and we will find the solution for your query and answer you. Thanks Join Our Telegram Group Join Our WhatsApp Group Sometimes we want to merge multiple sheets into one sheet so that we can easily analyse the data and turn it into some useful information. This articles will tell you how to merge multiple worksheets into one worksheet using VBA.Example:Here I have fetched some data from server that returns data into different worksheets. I have added one more sheet and named it as Master. Other sheet names doesnt matter.Now run this macro.Sub Merge_Sheets()Dim startRow, startCol, lastRow, lastCol As LongDim headers As Range'Set Master sheet for consolidationSet mtr = Worksheets("Master")Set wb = ThisWorkbook'Get HeadersSet headers = Application.InputBox("Select the Headers", Type:=8)'Copy Headers into masterheaders.Copy mtr.Range("A1")startRow = headers.Row + 1startCol = headers.ColumnDebug.Print startRow, startCol'loop through all sheetsFor Each ws In wb.Worksheets 'except the master sheet from looping If ws.Name "Master" Then ws.Activate lastRow = Cells(Rows.Count, startCol).End(xlUp).Row lastCol = Cells(startRow, Columns.Count).End(xlToLeft).Column 'get data from each worksheet and copy it into Master sheet Range(Cells(startRow, startCol), Cells(lastRow, lastCol)).Copy _ mtr.Range("A" & mtr.Cells(Rows.Count, 1).End(xlUp).Row + 1) End IfNext wsWorksheets("Master").ActivateEnd SubHow to merge sheets using this VBA Macro?Insert a new sheet and name it Master in the workbook. Rename it later if you want.Insert a module in VBA editor and copy above VBA code.Run the macro.You will be asked to select headings. Select the heading and hit OK.And it is done. All the sheets are merged in master.How it works?I assume that you know the basics of object and variable creation in VBA. in the first part we have created object and variables that we will need in our operations.Well most of the things I have explained using comments in vba code. Lets look at the main part of this vba code.For Each ws In wb.Worksheets 'except the master sheet from looping If ws.Name "Master" Then ws.Activate lastRow = Cells(Rows.Count, startCol).End(xlUp).Row lastCol = Cells(startRow, Columns.Count).End(xlToLeft).Column 'get data from each worksheet and copy it into Master sheet Range(Cells(startRow, startCol), Cells(lastRow, lastCol)).Copy _ mtr.Range("A" & mtr.Cells(Rows.Count, 1).End(xlUp).Row + 1) End IfNext wsIn earlier articles we learned how to loop through sheets and how to get last row and column using vba. Here we are looping through each sheet in main workbook using for loop.For Each ws In wb.WorksheetsThen we exclude master sheet from looping, since we will be consolidating our data in that sheet.Then we get last row and last column number. Now next line is very important. We have done multiple operations into one line.Range(Cells(startRow, startCol), Cells(lastRow, lastCol)).Copy _ mtr.Range("A" & mtr.Cells(Rows.Count, 1).End(xlUp).Row + 1)First we form a range using startRow, startCol and lastRow and lastCol. Range(Cells(startRow, startCol), Cells(lastRow, lastCol))We have copied it using copy method of range.Range(Cells(startRow, startCol), Cells(lastRow, lastCol)).CopyWe pasted it directly into first blank cell after last non blank cell in column A of master sheet (mtr.Cells(Rows.Count, 1).End(xlUp).Row + 1).Range(Cells(startRow, startCol), Cells(lastRow, lastCol)).Copy _ mtr.Range("A" & mtr.Cells(Rows.Count, 1).End(xlUp).Row + 1)This loops runs for all the sheets and copies each sheets data into master sheet.Finally, in the end of the macro we activate the mastersheet to see the output.So yeah guys, this is how you can merge every sheet in a workbook. Let me know if you have any query regarding this VBA code or any excel topic in the comments section below.Download file: Consolidate_Merge multiple worksheets into one master sheet using VBARelated Articles:How to loop through sheetshow to get last row and column using vbaDelete sheets without confirmation prompts using VBA in Microsoft ExcelAdd And Save New Workbook Using VBA In Microsoft Excel 2016Display A Message On The Excel VBA Status BarTurn Off Warning Messages Using VBA In Microsoft Excel 2016Popular Articles:The VLOOKUP Function in ExcelCOUNTIF in Excel 2016How to Use SUMIF Function in Excel Copy the cell ranges that you want to move into the master workbook. You can press Ctrl + C after selecting the cell range or go to the Home tab and click on the Copy command.Click on the plus icon at the bottom of your workbook.Press Ctrl + V to paste the data into the master workbook. Method 2 Copy the Worksheets to Combine Multiple Workbooks into One Workbook in ExcelSelect all the worksheets that you want to move or copy into another master worksheet. You can hold the Ctrl key and manually select all the worksheets using your mouse.Right-click on the selection area and a list will pop up.Choose Move or Copy from the list.The Move or Copy dialog box will appear. Select the workbook where you want to move or copy. This option is available in the To Book drop-down box.Select the position of the copied or moved worksheets in the new workbook under the Before Sheet command text.The selected worksheets are moved into the new workbook by default. If you want to copy them, check the Create a copy check box.Hit the OK button. Method 3 Use the INDIRECT Function to Combine Multiple Workbooks into One Workbook in ExcelInsert the File Name, Sheet Name, and Cell addresses from where you want to retrieve data in the destination worksheet.Insert the following formula in the destination cell address of the destination workbook: =INDIRECT("'[" & B3 & "]" & C3 & "'!" & D3) B3 contains the source Excel file name.C3 holds the worksheet name.D3 refers to the cell address of the source file from where you want to pull data.Press the Enter key.Drag the Fill Handle icon to the cell ranges where you want to get the data from the source file. Method 4 Use the CSV File Format to Combine Multiple Workbooks into One Workbook in ExcelSave each of the Excel files in CSV file format.Store them all in the same folder.Open the Command Prompt.Navigate to the folder where youve stored all the CSV files.Enter the following command. Here, Combined is an arbitrary file name. You can choose whatever you like. Open the Combined.csv file.Save as a normal Excel file (.xlsx). Method 5 Use PowerQuery to Combine Multiple Workbooks in One Workbook in ExcelStore all the Excel files in the same folder to combine them all together.Go to the Data tab from the main ribbon.Choose the Get Data command.Navigate to From File and to From Folder.Browse the folder where you stored all the Excel files.Select them all to combine into one master workbook. Method 6 Use VBA Code to Combine Multiple Workbooks into One Workbook in ExcelPress Alt + F11 to open the VBA editor.Go to Insert and selectModule to create a new module.Copy the following VBA code and paste it into the VBA editor.Sub CombineMultipleFiles()Dim x, i As IntegerDim y As FileDialogDim m, n As WorkbookDim z As WorksheetSet m = Application.ActiveWorkbookSet y = Application.FileDialog(msoFileDialogFilePicker)y.AllowMultiSelect = Truex = y.ShowFor i = 1 To y.SelectedItems.CountWorkbooks.Open y.SelectedItems(i)Set n = ActiveWorkbookFor Each z In n.Worksheetsz.Copy after:=m.Sheets(m.Worksheets.Count)Next zn.CloseNext iEnd SubSave the code with Ctrl + S.Hit the Run Sub button or press the F5 key to run the code.You will get prompts to select files. Pick as many Excel files as you want to combine them all together. Download the Practice Workbook Combine Multiple Workbooks to One.xlsm