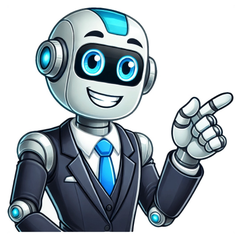
Click to verify



Posted by: | On: | “`html Microsoft Excel is a powerful tool that offers a multitude of features beyond basic spreadsheet capabilities. One of the most beneficial features for data analysis is the ability to automate tasks using Visual Basic for Applications (VBA). In this post, we will delve into the VBA command for calculating the average of a set of numbers, known as the Average function. We will cover the basics, show you how to use it, and provide examples to illustrate its practical applications. Understanding the Average Function in VBA The Average function in VBA is used to calculate the mean of a group of numbers. It is a straightforward function that simplifies the process of finding the average, especially when dealing with large datasets. This function is particularly useful in automating repetitive tasks that involve statistical analysis. What is the Average Function? The Average function computes the arithmetic mean of a specified range of cells or numbers. In other words, it sums up all the values and then divides the total by the number of values. This function is built into Excel VBA, making it easily accessible for users to perform calculations without manual input. How to Use the Average Function in Excel VBA Using the Average function within Excel’s VBA environment is simple once you understand the syntax and method of implementation. Below is a step-by-step guide on how to effectively use this function in your Excel projects. Step 1: Access the VBA Editor To use the Average function, you first need to access the VBA Editor. This is done by pressing Alt + F11 in Excel. The VBA Editor is where you will write your code and automate tasks. Step 2: Write the VBA Code Below is a basic example of how to use the Average function in VBA: Sub CalculateAverage() Dim rng As Range Dim avg As Double ' Define the range of cells Set rng = Range("A1:A10") ' Calculate the average avg = Application.WorksheetFunction.Average(rng) ' Display the result MsgBox "The average is " & avg End Sub In this example, we define a range of cells from A1 to A10 and use the Application.WorksheetFunction.Average method to calculate the average. The result is then displayed in a message box. Step 3: Run the Macro After writing the code, you can run the macro by returning to the Excel window, pressing Alt + F8, selecting the CalculateAverage macro, and clicking Run. This will execute the code and show you the average of the specified range. Practical Examples of the Average Function Understanding how to use the Average function is crucial, but seeing it applied in real-world scenarios can help solidify your understanding. Let’s explore a few practical examples. Example 1: Automating Monthly Sales Analysis Suppose you manage a team of sales representatives, and you want to automate the calculation of their average monthly sales. Here’s how you could set it up in VBA: Sub SalesAverage() Dim salesRange As Range Dim salesAvg As Double ' Define the range of sales data Set salesRange = Range("B2:B13") ' Calculate the average sales salesAvg = Application.WorksheetFunction.Average(salesRange) ' Output the result in a cell Range("B15").Value = salesAvg End Sub In this scenario, the sales data is located in the range B2:B13, and the calculated average is output to cell B15. Example 2: Average Student Grades In educational settings, you might need to quickly calculate the average grades of students. The following VBA script demonstrates this: Sub AverageGrades() Dim gradesRange As Range Dim gradesAvg As Double ' Define the range of student grades Set gradesRange = Range("C2:C20") ' Calculate the average grade gradesAvg = Application.WorksheetFunction.Average(gradesRange) ' Display the result in a message box MsgBox "The average grade is " & gradesAvg End Sub This script calculates the average from grades located in cells C2 to C20 and displays the result in a message box. Conclusion The Average function in Excel VBA is a powerful tool that can save time and improve accuracy in data analysis. By understanding how to implement this function, you can automate complex tasks and streamline your workflow. Whether you’re analyzing sales data, calculating grades, or working with any other numerical data, the Average function is an invaluable asset in your VBA toolkit. For those looking to further enhance their VBA skills, consider exploring more advanced functions and automation techniques. Check out our advanced VBA techniques guide for more insights. To learn more about Excel’s built-in functions, you can visit the official Microsoft Excel support page. “` This tutorial will demonstrate how to use the Excel Average function in VBA. The Excel AVERAGE Function is used to calculate an average from a range cells in your Worksheet that have values in them. In VBA, It is accessed using the WorksheetFunction method. AVERAGE WorksheetFunction The WorksheetFunction object can be used to call most of the Excel functions that are available within the Insert Function dialog box in Excel. The AVERAGE function is one of them. Sub TestFunction Range("D33") = Application.WorksheetFunction.Average("D1:D32") End Sub You are able to have up to 30 arguments in the AVERAGE function. Each of the arguments must refer to a range of cells. This example below will produce the average of the sum of the cells B11 to N11 Sub TestAverage() Range("O11") = Application.WorksheetFunction.Average(Range("B11:N11")) End Sub The example below will produce an average of the sum of the cells in B11 to N11 and the sum of the cells in B12:N12. If you do not type the Application object, it will be assumed. Sub TestAverage() Range("O11") = WorksheetFunction.Average(Range("B11:N11"),Range("B12:N12")) End Sub Assign AVERAGE Result to a Variable You may want to use the result of your formula elsewhere in code rather than writing it directly back to an Excel Range. If this is the case, you can assign the result to a variable to use later in your code. Sub AssignAverage() Dim result As Integer 'Assign the variable result = WorksheetFunction.Average(Range("A10:N10")) 'Show the result MsgBox "The average for the cells in this range is " & result End Sub AVERAGE with a Range Object You can assign a group of cells to the Range object, and then use that Range object with the WorksheetFunction object. Sub TestAverageRange() Dim rng As Range 'assign the range of cells Set rng = Range("G2:G7") 'use the range in the formula Range("G8") = WorksheetFunction.Average(rng) 'release the range object Set rng = Nothing End Sub AVERAGE Multiple Range Objects Similarly, you can calculate the average of the cells from multiple Range Objects. Sub TestAverageMultipleRanges() Dim rngA As Range Dim rngB as Range 'assign the range of cells Set rngA = Range("D2:D10") Set rngB = Range("E2:E10") 'use the range in the formula Range("E11") = WorksheetFunction.Average(rngA, rngB) 'release the range object Set rngA = Nothing Set rngB = Nothing End Sub Using AVERAGEA The AVERAGEA Function differs from the AVERAGE function in that it create an average from all the cells in a range, even if one of the cells has text in it – it replaces the text with a zero and includes that in calculating the average. The AVERAGE function would ignore that cell and not factor it into the calculation. Sub TestAverageA() Range("B8") = Application.WorksheetFunction.AverageA(Range("A10:A11")) End Sub In the example below, the AVERAGE function returns a different value to the AVERAGEA function when the calculation is used on cells A10 to A11 The answer for the AVERAGEA formula is lower than the AVERAGE formula as it replaces the text in A11 with a zero, and therefore averages over 13 values rather than the 12 values that the AVERAGE is calculating over. Using AVERAGEIF The AVERAGEIF Function allows you to average the sum of a range of cells that meet a certain criteria. Sub AverageIf() Range("F31") = WorksheetFunction.AverageIf(Range("F5:F30"), "Savings", Range("G5:G30")) End Sub The procedure above will only average the cells in range G5:G30 where the corresponding cell in column F has the word ‘Savings’ in it. The criteria you use has to be in quotation marks. Disadvantages of WorksheetFunction When you use the WorksheetFunction to average the values in a range in your worksheet, a static value is returned, not a flexible formula. This means that when your figures in Excel change, the value that has been returned by the WorksheetFunction will not change. In the example above, the procedure TestAverage procedure has created the average of B11:M11 and put the answer in N11. As you can see in the formula bar, this result is a figure and not a formula. If any of the values change therefore in the Range(B11:M11), the results in N11 will NOT change. Instead of using the WorksheetFunction.Average, you can use VBA to apply the AVERAGE Function to a cell using the Formula or FormulaR1C1 methods. Using the Formula Method The formula method allows you to point specifically to a range of cells eg: B11:M11 as shown below. Sub TestAverageFormula() Range("N11").Formula = "=Average(B11:M11)" End Sub Using the FormulaR1C1 Method The FomulaR1C1 method is more flexible in that it does not restrict you to a set range of cells. The example below will give us the same answer as the one above. Sub TestAverageFormula() Range("N11").Formula = "=Average(RC[-12]:RC[-1])" End Sub However, to make the formula more flexible, we could amend the code to look like this: Sub TestAverageFormula() ActiveCell.FormulaR1C1 = "=Average(R[-11]C:R[-1]C)" End Sub Wherever you are in your worksheet, the formula will then average the values in the 12 cells directly to the left of it and place the answer into your ActiveCell. The Range inside the AVERAGE function has to be referred to using the Row (R) and Column (C) syntax. Both these methods enable you to use Dynamic Excel formulas within VBA. There will now be a formula in N11 instead of a value. VBA (Visual Basic for Applications) is the programming language of Excel and other offices. It is an event-driven programming language from Microsoft. With Excel VBA, one can automate many tasks in excel and all other office softwares. It helps in generating reports, preparing various charts, graphs and moreover, it performs calculation using its various functions. Let's see Average functions in Excel. AVERAGE It returns the arithmetic mean of all of its arguments. Syntax: =AVERAGE(number1, number2, ...) Here, number 1: This is the first number in your cell. You can specify it upto 255 numbers. number 2: This is an Optional field. You can specify it upto 255 numbers. Example: Output: AVERAGEIF: It calculates the average of only those values which meet a certain criteria. Syntax: =AVERAGEIF(range, criteria, average_range) range: It is the range of cells to be evaluated. The cells in range can be numbers, names, arrays or any other reference that contains numbers. Blank and text values are not considered. criteria: It should be in the form of a number or any expression which defines which all cells to be averaged. For instance, “mango”, C6, “ If a cell in average_range is an empty cell, it will be ignored. -> If range is a text value or blank, it will return an error. -> If a cell in criteria is empty, it will be considered as 0 value. -> If no cell meets the criteria, it will return an error. Example: Output: AVERAGEIFS: Calculate average of cells that meet multiple criteria. Syntax: =AVERAGEIFS(average_range, criteria_range1, criteria1, [criteria_range2, criteria2], ...) average_range: (Required) One or more cells to calculate average. criteria_range1: (Required) 1 to 127 ranges to evaluate the given criteria. criteria1: 1 to 127 criteria in the form of numbers, expression to be averaged. This defines which all cells in criteria_range1 are to be averaged. For instance, criteria can be like this, “>32”, “apple”, “D7” Example: Output: This tutorial will demonstrate how to use the Excel Average function in VBA. The Excel AVERAGE Function is used to calculate an average from a range cells in your Worksheet that have values in them. In VBA, It is accessed using the WorksheetFunction method. AVERAGE WorksheetFunction The WorksheetFunction object can be used to call most of the Excel functions that are available within the Insert Function dialog box in Excel. The AVERAGE function is one of them. Sub TestFunction Range("D33") = Application.WorksheetFunction.Average("D1:D32") End Sub You are able to have up to 30 arguments in the AVERAGE function. Each of the arguments must refer to a range of cells. This example below will produce the average of the sum of the cells B11 to N11 Sub TestAverage() Range("O11") = Application.WorksheetFunction.Average(Range("B11:N11")) End Sub The example below will produce an average of the sum of the cells in B11 to N11 and the sum of the cells in B12:N12. If you do not type the Application object, it will be assumed. Sub TestAverage() Range("O11") = WorksheetFunction.Average(Range("B11:N11"),Range("B12:N12")) End Sub Assign AVERAGE Result to a Variable You may want to use the result of your formula elsewhere in code rather than writing it directly back to an Excel Range. If this is the case, you can assign the result to a variable to use later in your code. Sub AssignAverage() Dim result As Integer 'Assign the variable result = WorksheetFunction.Average(Range("A10:N10")) 'Show the result MsgBox "The average for the cells in this range is " & result End Sub AVERAGE with a Range Object You can assign a group of cells to the Range object, and then use that Range object with the WorksheetFunction object. Sub TestAverageRange() Dim rng As Range 'assign the range of cells Set rng = Range("G2:G7") 'use the range in the formula Range("G8") = WorksheetFunction.Average(rng) 'release the range object Set rng = Nothing End Sub AVERAGE Multiple Range Objects Similarly, you can calculate the average of the cells from multiple Range Objects. Sub TestAverageMultipleRanges() Dim rngA As Range Dim rngB as Range 'assign the range of cells Set rngA = Range("D2:D10") Set rngB = Range("E2:E10") 'use the range in the formula Range("E11") = WorksheetFunction.Average(rngA, rngB) 'release the range object Set rngA = Nothing Set rngB = Nothing End Sub Using AVERAGEA The AVERAGEA Function differs from the AVERAGE function in that it create an average from all the cells in a range, even if one of the cells has text in it – it replaces the text with a zero and includes that in calculating the average. The AVERAGE function would ignore that cell and not factor it into the calculation. Sub TestAverageA() Range("B8") = Application.WorksheetFunction.AverageA(Range("A10:A11")) End Sub In the example below, the AVERAGE function returns a different value to the AVERAGEA function when the calculation is used on cells A10 to A11 The answer for the AVERAGEA formula is lower than the AVERAGE formula as it replaces the text in A11 with a zero, and therefore averages over 13 values rather than the 12 values that the AVERAGE is calculating over. Using AVERAGEIF The AVERAGEIF Function allows you to average the sum of a range of cells that meet a certain criteria. Sub AverageIf() Range("F31") = WorksheetFunction.AverageIf(Range("F5:F30"), "Savings", Range("G5:G30")) End Sub The procedure above will only average the cells in range G5:G30 where the corresponding cell in column F has the word ‘Savings’ in it. The criteria you use has to be in quotation marks. Disadvantages of WorksheetFunction When you use the WorksheetFunction to average the values in a range in your worksheet, a static value is returned, not a flexible formula. This means that when your figures in Excel change, the value that has been returned by the WorksheetFunction will not change. In the example above, the procedure TestAverage procedure has created the average of B11:M11 and put the answer in N11. As you can see in the formula bar, this result is a figure and not a formula. If any of the values change therefore in the Range(B11:M11), the results in N11 will NOT change. Instead of using the WorksheetFunction.Average, you can use VBA to apply the AVERAGE Function to a cell using the Formula or FormulaR1C1 methods. Using the Formula Method The formula method allows you to point specifically to a range of cells eg: B11:M11 as shown below. Sub TestAverageFormula() Range("N11").Formula = "=Average(B11:M11)" End Sub Using the FormulaR1C1 Method The FomulaR1C1 method is more flexible in that it does not restrict you to a set range of cells. The example below will give us the same answer as the one above. Sub TestAverageFormula() Range("N11").Formula = "=Average(RC[-12]:RC[-1])" End Sub However, to make the formula more flexible, we could amend the code to look like this: Sub TestAverageFormula() ActiveCell.FormulaR1C1 = "=Average(R[-11]C:R[-1]C)" End Sub Wherever you are in your worksheet, the formula will then average the values in the 12 cells directly to the left of it and place the answer into your ActiveCell. The Range inside the AVERAGE function has to be referred to using the Row (R) and Column (C) syntax. Both these methods enable you to use Dynamic Excel formulas within VBA. There will now be a formula in N11 instead of a value. This tutorial will demonstrate how to use the Excel Average function in VBA. The Excel AVERAGE Function is used to calculate an average from a range cells in your Worksheet that have values in them. In VBA, It is accessed using the WorksheetFunction method. AVERAGE WorksheetFunction The WorksheetFunction object can be used to call most of the Excel functions that are available within the Insert Function dialog box in Excel. The AVERAGE function is one of them. Sub TestFunction Range("D33") = Application.WorksheetFunction.Average("D1:D32") End Sub You are able to have up to 30 arguments in the AVERAGE function. Each of the arguments must refer to a range of cells. This example below will produce the average of the sum of the cells B11 to N11 Sub TestAverage() Range("O11") = Application.WorksheetFunction.Average(Range("B11:N11")) End Sub The example below will produce an average of the sum of the cells in B11 to N11 and the sum of the cells in B12:N12. If you do not type the Application object, it will be assumed. Sub TestAverage() Range("O11") = WorksheetFunction.Average(Range("B11:N11"),Range("B12:N12")) End Sub Assign AVERAGE Result to a Variable You may want to use the result of your formula elsewhere in code rather than writing it directly back to an Excel Range. If this is the case, you can assign the result to a variable to use later in your code. Sub AssignAverage() Dim result As Integer 'Assign the variable result = WorksheetFunction.Average(Range("A10:N10")) 'Show the result MsgBox "The average for the cells in this range is " & result End Sub AVERAGE with a Range Object You can assign a group of cells to the Range object, and then use that Range object with the WorksheetFunction object. Sub TestAverageRange() Dim rng As Range 'assign the range of cells Set rng = Range("G2:G7") 'use the range in the formula Range("G8") = WorksheetFunction.Average(rng) 'release the range object Set rng = Nothing End Sub AVERAGE Multiple Range Objects Similarly, you can calculate the average of the cells from multiple Range Objects. Sub TestAverageMultipleRanges() Dim rngA As Range Dim rngB as Range 'assign the range of cells Set rngA = Range("D2:D10") Set rngB = Range("E2:E10") 'use the range in the formula Range("E11") = WorksheetFunction.Average(rngA, rngB) 'release the range object Set rngA = Nothing Set rngB = Nothing End Sub Using AVERAGEA The AVERAGEA Function differs from the AVERAGE function in that it create an average from all the cells in a range, even if one of the cells has text in it – it replaces the text with a zero and includes that in calculating the average. The AVERAGE function would ignore that cell and not factor it into the calculation. Sub TestAverageA() Range("B8") = Application.WorksheetFunction.AverageA(Range("A10:A11")) End Sub In the example below, the AVERAGE function returns a different value to the AVERAGEA function when the calculation is used on cells A10 to A11 The answer for the AVERAGEA formula is lower than the AVERAGE formula as it replaces the text in A11 with a zero, and therefore averages over 13 values rather than the 12 values that the AVERAGE is calculating over. Using AVERAGEIF The AVERAGEIF Function allows you to average the sum of a range of cells that meet a certain criteria. Sub AverageIf() Range("F31") = WorksheetFunction.AverageIf(Range("F5:F30"), "Savings", Range("G5:G30")) End Sub The procedure above will only average the cells in range G5:G30 where the corresponding cell in column F has the word ‘Savings’ in it. The criteria you use has to be in quotation marks. Disadvantages of WorksheetFunction When you use the WorksheetFunction to average the values in a range in your worksheet, a static value is returned, not a flexible formula. This means that when your figures in Excel change, the value that has been returned by the WorksheetFunction will not change. In the example above, the procedure TestAverage procedure has created the average of B11:M11 and put the answer in N11. As you can see in the formula bar, this result is a figure and not a formula. If any of the values change therefore in the Range(B11:M11), the results in N11 will NOT change. Instead of using the WorksheetFunction.Average, you can use VBA to apply the AVERAGE Function to a cell using the Formula or FormulaR1C1 methods. Using the Formula Method The formula method allows you to point specifically to a range of cells eg: B11:M11 as shown below. Sub TestAverageFormula() Range("N11").Formula = "=Average(B11:M11)" End Sub Using the FormulaR1C1 Method The FomulaR1C1 method is more flexible in that it does not restrict you to a set range of cells. The example below will give us the same answer as the one above. Sub TestAverageFormula() Range("N11").Formula = "=Average(RC[-12]:RC[-1])" End Sub However, to make the formula more flexible, we could amend the code to look like this: Sub TestAverageFormula() ActiveCell.FormulaR1C1 = "=Average(R[-11]C:R[-1]C)" End Sub Wherever you are in your worksheet, the formula will then average the values in the 12 cells directly to the left of it and place the answer into your ActiveCell. The Range inside the AVERAGE function has to be referred to using the Row (R) and Column (C) syntax. Both these methods enable you to use Dynamic Excel formulas within VBA. There will now be a formula in N11 instead of a value. You can use the following methods to write AVERAGEIF and AVERAGEIFS functions using VBA in Excel: Method 1: AVERAGEIF Function in VBA Sub Averageif_Function() Range("E2") = WorksheetFunction.AverageIf(Range("A2:A12"), "Mavs", Range("B2:B12")) End Sub This particular example will calculate the average value in the range B2:B12 where the corresponding value in the range A2:A12 is equal to “Mavs” and then assign the result to cell E2. Method 2: AVERAGEIFS Function in VBA Sub Averageifs_Function() Range("E2") = WorksheetFunction.AverageIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20") End Sub This particular example will calculate the average value in the range C2:C12 where the value in the range A2:A12 is equal to “Mavs” and the value in the range B2:B12 is greater than 20 and then assign the result to cell E2. The following examples shows how to use each of these methods in practice with the following dataset in Excel that contains information about various basketball players: Example 1: AVERAGEIF Function in VBA Suppose we would like to calculate the average value in the points column for players who are on the Mavs team. We can create the following macro to perform this AVERAGEIF function: Sub Averageif_Function() Range("E2") = WorksheetFunction.AverageIf(Range("A2:A12"), "Mavs", Range("B2:B12")) End Sub When we run this macro, we receive the following output: Notice that cell E2 contains a value of 18.25. This tells us that the average value in the points column for the players on the Mavs team is 18.25. We can verify that this is correct by manually calculating the average points value for the players on the Mavs: Average points for Mavs players: (22 + 10 + 29 + 12) / 4 = 18.25. Example 2: AVERAGEIFS Function in VBA Suppose we would like to calculate the average value in the assists column for the players that meet the following criteria: Player is on the Mavs team. Player scored more than 20 points. We can create the following macro to perform this AVERAGEIFS function: Sub Averageifs_Function() Range("E2") = WorksheetFunction.AverageIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20") End Sub When we run this macro, we receive the following output: Notice that cell E2 contains a value of 9. This tells us that the average assists among players who are on the Mavs team and scored more than 20 points is 9. Note: In this example, we created a AVERAGEIFS function using two criteria ranges but you can use as many criteria ranges as you’d like within the WorksheetFunction.AverageIfs method. Additional Resources The following tutorials explain how to perform other common tasks in VBA: VBA: How to Calculate Average Value of Range VBA: How to Write SUMIF and SUMIFS Functions VBA: How to Write COUNTIF and COUNTIFS Functions Quick View: Function TopAverage(DataAvg, Num) 'Returns the average of the highest Num values in Data Sum = 0 For i = 1 To Num Sum = Sum + WorksheetFunction.Large(DataAvg, i) 'Large(Data, i) returns the ith largest value from the Data. Next i TopAverage = Sum / Num MsgBox "The average is: " & TopAverage End Function The VBA Average Function in Excel → Function Objective calculate the arithmetic mean. → Syntax expression.Average(Arg1,Arg2…) → Arguments ARGUMENT REQUIRED/OPTIONAL Data Type EXPLANATION Arg1,Arg2… Required Variant Numeric value to calculate the average This is the sample dataset. Example 1 – Apply the VBA Average Function to Calculate the Average of an Array Steps: Developer → Visual Basic Select Visual Basic. Go to: Insert → Module In the VBA Average Function module, use the VBA code. Sub Average_Array() Range("D13").Formula = "=Average(D5:D12)" End Sub To run the VBA code, click: Run → Run Sub/UserForm Go back to the Excel sheet. This is the output: Read More: How to Calculate Average of Multiple Columns in Excel Example 2 – Use the VBA Average Function with a Range Object Steps: Follow the steps described in Method 1 to insert a new module. Enter the VBA code to calculate the average of the range object. Sub Average_Range_Object() Dim x As Range Set x = Range("D5:D12") Range("D13") = WorksheetFunction.Average(x) Set x = Nothing End Sub To run the VBA code, click: Run → Run Sub/UserForm Go back to the Excel sheet. The average is $27,780.00. Read More: How to Calculate Average of Multiple Ranges in Excel Example 3 – Apply a Custom Function to Calculate the Average in Excel VBA Steps: Follow the steps described in Method 1 to insert a new module. Enter the VBA code. Function TopAverage(DataAvg, Num) 'Returns the average of the highest Num values in Data Sum = 0 For i = 1 To Num Sum = Sum + WorksheetFunction.Large(DataAvg, i) 'Large(Data, i) returns the ith largest value from the Data. Next i TopAverage = Sum / Num MsgBox "The average is: " & TopAverage End Function Press Ctrl + S. Go back to the dataset and select D13. Enter the function: D5:D12 is the DataAvg, and 5 the Num of the TopAverage function. A message box displays: “The average is: 35723.8”. Example 4 – Use the VBA Average Function to Calculate the Average of a Dynamic Range of Cells Steps: Follow the steps described in Method 1 to insert a new module. Enter the VBA code. Sub Average_Function() Dim R As Double R = Evaluate("AVERAGE(D5:D" & Sheets("Dynamic").Range("D" & Rows.Count).End(xlUp).Row & ")") MsgBox "The Average of Dynamic Range is: $" & R End Sub To run the VBA code, click: Run → Run Sub/UserForm A message box displays “The Average of Dynamic range is: $27780” . Things to Remember You can open the Microsoft Visual Basic for Applications window by pressing Alt + F11. To enable the Developer tab, go to: File → Option → Customize Ribbon Download Practice Workbook Download the practice workbook. VBA Average in Excel.xlsm Related Articles