How to Easily Write SUMIF and SUMIFS Functions in VBA

Authored by

stats writer

November 20, 2025

The ability to calculate conditional sums is fundamental for data analysis in Microsoft Excel. When automating complex reports or large-scale data processing, embedding these calculations directly into VBA becomes essential. The **SUMIF** and **SUMIFS** functions are powerful tools used within VBA to quickly aggregate numerical values based on specific, user-defined criteria.

The core difference between the two functions lies in their complexity: the SUMIF Function is designed to calculate the sum of cells based on a **single criterion**, whereas the SUMIFS Function handles calculations requiring **multiple simultaneous criteria** to be met. Both functions are accessed in VBA using the `WorksheetFunction` object, which allows developers to leverage Excel's built-in calculation engine within their code. For situations demanding even more flexible array-based calculations, developers may also consider integrating the SUMPRODUCT function.

Accessing Excel Functions via VBA's WorksheetFunction Object

To implement functions like SUMIF or SUMIFS directly within VBA code, we must utilize the **WorksheetFunction** object. This object acts as a bridge, making nearly all standard Excel worksheet functions available for use within your procedural code. When using these functions in VBA, it is crucial to remember that they expect arguments to be passed as VBA objects, typically `Range` objects, rather than simple cell references used in spreadsheet formulas.

When you write a formula in a spreadsheet, you might type `=SUMIF(A1:A10, "Criteria", B1:B10)`. However, in VBA, this translates to using the `WorksheetFunction` method followed by the function name, passing `Range` objects as parameters. Errors in argument types or ranges will often result in runtime errors that halt the execution of the subroutine.

Syntax and Arguments of the SUMIF Function

The SUMIF Function is the simplest way to perform conditional summation in VBA, requiring three primary arguments. Understanding the order and purpose of these arguments is key to successful implementation. The arguments are, in order: the **Range** containing the criteria, the **Criteria** itself (which can be a numerical value, text string, or expression), and the **Sum_Range** (the actual cells whose values will be added up).

When structuring the VBA code, the calculation is typically assigned directly to a specific cell or variable. This means the result of the `WorksheetFunction.SumIf` operation is immediately placed where it is needed, bypassing the need to write the formula directly onto the worksheet first. This efficient methodology is one of the primary benefits of using VBA for complex calculations.

You can use the following methods to write SUMIF and SUMIFS functions using VBA in Excel:

Method 1: Implementing the SUMIF Function in VBA

This method demonstrates a straightforward application of the SUMIF Function to sum values based on a single condition. Note the use of the `WorksheetFunction` object, which is necessary to call native Excel functions from within a VBA subroutine. The code structure involves defining a subroutine, performing the calculation, and assigning the output to a target cell.

```
Sub Sumif_Function()
Range("E2") = WorksheetFunction.Sumif(Range("A2:A12"), "Mavs", Range("B2:B12"))
End Sub
```

In this particular example, the code is instructed to sum the values found in the range **B2:B12**. This summation only occurs where the corresponding value in the criteria range **A2:A12** precisely matches the text string "Mavs". The final calculated total is then directly assigned to the target cell **E2** on the active worksheet, completing the operation efficiently within the executed VBA macro.

Syntax and Arguments of the SUMIFS Function

Unlike its single-criterion counterpart, the SUMIFS Function is structured differently to accommodate multiple logical tests. Crucially, the order of arguments is inverted compared to SUMIF. The first argument in SUMIFS must be the **Sum_Range**, followed by pairs of **Criteria_Range** and **Criteria** arguments. This sequence allows the function to process an unlimited number of criteria pairs, provided the maximum argument limit is not exceeded.

The logic employed by SUMIFS is strictly based on the AND condition. This means that a row's value will only be included in the total sum if **all** specified criteria are met simultaneously across their respective ranges. This makes SUMIFS indispensable for highly specific data filtering and aggregation tasks where complex conditional logic is required before summing values.

Method 2: Implementing the SUMIFS Function in VBA

This example illustrates how to utilize the SUMIFS Function to apply two separate criteria before summing the desired values. Notice that the sum range, `C2:C12`, is declared first, adhering to the required syntax of the SUMIFS worksheet function.

```
Sub Sumifs_Function()
Range("E2") = WorksheetFunction.SumIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20")
End Sub
```

This sophisticated example performs summation over the range **C2:C12** only when two independent conditions are satisfied: first, the corresponding value in range **A2:A12** must be exactly equal to "Mavs," *and* second, the value in range **B2:B12** must be greater than 20. Only rows meeting both conditions contribute to the sum, which is then written into cell **E2**.

Illustrative Dataset for Examples

The following examples demonstrate how to use each of these conditional summation methods in practice, utilizing a consistent dataset in Excel. This data contains statistics for various basketball players, allowing us to filter sums based on team affiliation and performance metrics:

	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	4			
3	Mavs	10	6			
4	Warriors	14	8			
5	Hawks	15	10			
6	Mavs	29	14			
7	Kings	34	13			
8	Kings	30	5			
9	Hawks	29	8			
10	Kings	24	10			
11	Warriors	15	4			
12	Mavs	12	9			
13						
14						
15						
16						
17						
18						
19						
20						

Detailed Example 1: Implementing SUMIF for Single Criteria

For our first practical demonstration, assume we are interested in calculating the total offensive output, specifically the sum of all values in the Points column, exclusively for players associated with the "Mavs" team. This task requires filtering the data based on a single criterion found in the Team column, making the SUMIF Function the ideal choice.

We will construct a dedicated VBA macro to execute this conditional summation. The macro

targets the team names in column A (A2:A12), searches for "Mavs," and sums the corresponding values in column B (B2:B12, the Points column).

```
Sub Sumif_Function()
Range("E2") = WorksheetFunction.Sumif(Range("A2:A12"), "Mavs", Range("B2:B12"))
End Sub
```

Upon successfully executing this VBA subroutine, the targeted cell receives the cumulative total. The resulting output demonstrates how the calculation is performed invisibly by the code and immediately displayed on the sheet:

	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	4		73	
3	Mavs	10	6			
4	Warriors	14	8			
5	Hawks	15	10			
6	Mavs	29	14			
7	Kings	34	13			
8	Kings	30	5			
9	Hawks	29	8			
10	Kings	24	10			
11	Warriors	15	4			
12	Mavs	12	9			
13						
14						
15						
16						
17						
18						
19						
20						

Observation shows that cell **E2** now holds the value of **73**. This result accurately represents the consolidated sum of all Points scored by players affiliated with the Mavs team within the provided dataset. We can verify this calculation manually: Sum of Points for Mavs players: 22 + 10 + 29 + 12 = **73**.

Detailed Example 2: Implementing SUMIFS for Multiple Criteria

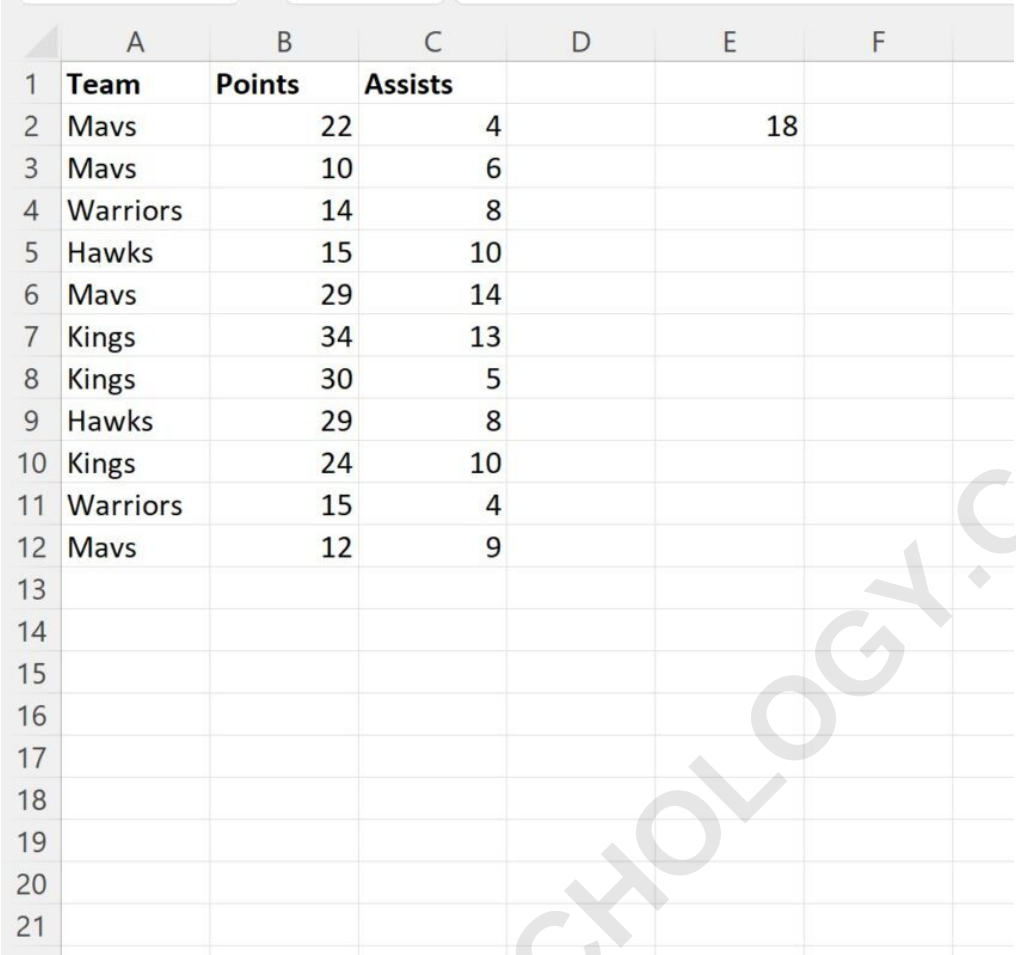
For our second example, we shift to a more complex scenario requiring the aggregation of assists based on two simultaneous conditions. We aim to calculate the sum of values in the Assists column (Column C) only for players who satisfy the following stringent requirements:

Player must belong to the **Mavs** team (Criteria 1).
Player must have scored **more than 20 points** (Criteria 2).

This necessitates the use of the SUMIFS Function to handle the intersection of these two criteria ranges. We will create the following macro to perform this multi-conditional calculation:

```
Sub Sumifs_Function()
Range("E2") = WorksheetFunction.SumIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20")
End Sub
```

Running this macro processes the complex criteria and updates the output cell. The resultant worksheet displays the calculated sum based on the combined conditions:



	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	4		18	
3	Mavs	10	6			
4	Warriors	14	8			
5	Hawks	15	10			
6	Mavs	29	14			
7	Kings	34	13			
8	Kings	30	5			
9	Hawks	29	8			
10	Kings	24	10			
11	Warriors	15	4			
12	Mavs	12	9			
13						
14						
15						
16						
17						
18						
19						
20						
21						

The updated cell **E2** now contains the value of **18**. This total represents the sum of Assists (Column C) exclusively for players who are Mavs AND scored over 20 points. Only two rows meet this combined criteria (the players scoring 22 and 29 points, with 5 and 13 assists, respectively: 5 + 13 = 18).

Conclusion and Advanced Considerations

Implementing conditional summation functions like SUMIF and SUMIFS within VBA provides significant advantages in terms of automation, speed, and clean code management. By employing the `WorksheetFunction` object, developers can tap into Excel's robust calculation engine without resorting to writing volatile formulas directly onto the sheet.

It is important to note that while this specific Example 2 used only two criteria ranges, the `WorksheetFunction.SumIfs` method is highly flexible. You can include as many criteria ranges and criteria pairs as needed to define complex data subsets for summation, making it a highly versatile tool for data extraction and reporting in Excel automation projects.