

How to SUMIF with Multiple Columns in Google Sheets: A Step-by-Step Guide

Authored by
stats writer

December 3, 2025

RECOMMENDED CITATION

stats writer (2025). *How to SUMIF with Multiple Columns in Google Sheets: A Step-by-Step Guide*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=104182>

Mastering complex data analysis in [Google Sheets](#) often requires more than simple summation. While the basic [SUMIF function](#) is highly effective for calculating sums based on a single condition, real-world datasets frequently demand calculations that adhere to several restrictions simultaneously. When you need to perform [data aggregation](#) across a range of cells where multiple criteria must be satisfied--such as summing sales figures for 'Region East' *and* 'Product A' *and* 'Status Complete'--you need a more robust tool.

This necessity leads us directly to the [SUMIFS function](#). Unlike its singular counterpart, [SUMIFS](#) is specifically designed to handle the logical intersection of multiple conditional ranges, allowing analysts to extract highly specific totals from large tables. Understanding how to deploy this function effectively is crucial for accurate reporting and insightful data interpretation within the Google Sheets environment, especially when dealing with data that spans across multiple criteria columns.

The Necessity of Multi-Criteria Summation in Data Analysis

The ability to sum data based on specific conditions, known as conditional summation, forms the bedrock of many analytical tasks. Data processing often involves filtering large amounts of information to isolate the relevant subset before calculating totals. While single-condition filtering provided by [SUMIF function](#) is straightforward, multi-condition filtering demands precise functional logic that combines requirements from different columns.

The standard [SUMIF function](#) is structured to evaluate a single criterion against a defined range and then sum corresponding values in a parallel range. While excellent for basic tasks, its limitation becomes apparent quickly: it cannot natively handle a scenario where the sum must depend on conditions met across three or four separate columns. Analysts previously had to resort to complex array formulas or helper columns to achieve multi-criteria summing, which often resulted in cumbersome and error-prone spreadsheets that were difficult to audit and maintain.

Fortunately, [Google Sheets](#) provides the specialized **SUMIFS()** function to address this exact requirement. This function streamlines the process by allowing the user to specify numerous criteria pairs--each consisting of a criteria range and the specific condition (or [criterion](#)) to be met--all within a single, elegant formula structure. This capability is indispensable when dealing with detailed transaction logs, performance metrics, or complex inventory databases where sums must satisfy several constraints simultaneously.

Decoding the SUMIFS Syntax and Structure

Understanding the proper syntax is the first step toward effectively utilizing **SUMIFS()**. Unlike the older SUMIF function, where the sum range is placed last, **SUMIFS()** requires the sum range to be specified first. This structural difference is critical for ensuring the formula calculates correctly,

especially when managing long lists of criteria, as it establishes the target values immediately.

The basic structure of the **SUMIFS()** function in [Google Sheets](#) is as follows:

SUMIFS(sum_range, criteria_range1, criterion1, , ...)

Let us break down each essential component of this syntax to clarify its role and proper usage in conditional summation:

sum_range: This is the mandatory first argument. It defines the actual range of cells that contain the values to be summed up. Importantly, the values in this range are only included in the total if **all** corresponding criteria are met across the criteria ranges.

criteria_range1: This is the first range of cells that the function will evaluate against the first condition. This range must be the same size and dimension as the **sum_range** and all subsequent criteria ranges specified.

criterion1: This is the specific condition that must be satisfied within the **criteria_range1**. The criterion can be a number, a text string (which must be enclosed in quotation marks, e.g., "Active"), a cell reference (e.g., A1), or a logical expression (e.g., ">50").

criteria_range2, criterion2, ...: These are optional but critical arguments that allow the function to include additional conditions. You can extend the formula with up to 127 criteria pairs, enabling extraordinarily precise data filtering and aggregation.

Practical Application: Example 1 with Character Criteria

To demonstrate the practical application of **SUMIFS()**, we will first explore a scenario involving solely categorical data. Imagine we are managing a spreadsheet that tracks the performance statistics of various basketball players. We need to aggregate total points based on two specific characteristics: their team affiliation and their assigned playing position. This requires matching text strings across two different criteria columns within the dataset.

Suppose we are working with the following dataset detailing player statistics, where the relevant data points are contained in columns C, D, and E:

	A	B	C	D	E
1	Team	Position	Points	Assists	
2	Spurs	Guard	31	6	
3	Spurs	Guard	19	11	
4	Spurs	Forward	24	5	
5	Spurs	Forward	20	6	
6	Spurs	Center	6	2	
7	Mavericks	Guard	35	15	
8	Mavericks	Guard	14	8	
9	Mavericks	Forward	16	8	
10	Mavericks	Forward	8	4	
11	Mavericks	Center	3	7	
12					
13					
14					
15					
16					
17					
18					
19					
20					

In this structure, Column E contains the Points scored, which serves as our **sum_range**. Column C lists the Team, and Column D lists the Position. Our objective is strictly defined: we want to calculate the total points scored only by players who belong to the "Spurs" team *and* simultaneously play the "Guard" position. This necessitates satisfying two distinct, text-based criteria simultaneously using the logical AND function implicit in **SUMIFS()**.

Executing Example 1: Summing Based on Team and Position

The formula must instruct Google Sheets to look at the points in Column E, but only where the corresponding row in Column C equals "Spurs" AND the corresponding row in Column D equals "Guard." Assuming the data begins in Row 2 and extends to Row 10, we structure the formula as follows:

```
=SUMIFS(E2:E10, C2:C10, "Spurs", D2:D10, "Guard")
```

This instruction breaks down the calculation into three distinct actions for the function:

The function first identifies the values it intends to sum: the range **E2:E10** (Points).

It applies the first filter: checking the range **C2:C10** (Team) for the exact text string "**Spurs**".

It applies the second filter: checking the range **D2:D10** (Position) for the exact text string "**Guard**".

The function evaluates each row independently. If a row satisfies both criteria (Team is "Spurs" AND Position is "Guard"), the corresponding value from the Points column (E) is accurately added to the running total. The resulting output of the spreadsheet calculation is visually represented below, confirming the successful execution:

	A	B	C	D	E	F
F2						
1	Team	Position	Points	Assists		
2	Spurs	Guard	31	6		44
3	Spurs	Guard	19	11		
4	Spurs	Forward	24	5		
5	Spurs	Forward	20	6		
6	Spurs	Center	6	2		
7	Mavericks	Guard	35	15		
8	Mavericks	Guard	14	8		
9	Mavericks	Forward	16	8		
10	Mavericks	Forward	8	4		
11	Mavericks	Center	3	7		
12						
13						
14						
15						
16						
17						

As confirmed by the formula's execution, the total points scored by players who are on the Spurs *and* have a Guard position is calculated precisely as **44**. This example clearly illustrates how **SUMIFS()** handles the logical AND condition imposed by multiple textual criteria across separate columns.

Example 2: Combining Character and Numeric Criteria

The true flexibility of **SUMIFS()** shines when dealing with heterogeneous criteria--specifically, combining text filtering with numeric thresholds. This capability is essential for generating performance reports that rely on both descriptive categories (like team name) and quantitative performance indicators (like assists or scores).

Let us use the same dataset but introduce a new complexity. Our analytical objective is now more nuanced: we want to sum the points scored by players who meet two conditions: they must be on the "Spurs" team (a character criteria) *and* they must have recorded more than 5 assists (a numeric/relational criteria). The Assists data is located in Column F.

	A	B	C	D	E
1	Team	Position	Points	Assists	
2	Spurs	Guard	31	6	
3	Spurs	Guard	19	11	
4	Spurs	Forward	24	5	
5	Spurs	Forward	20	6	
6	Spurs	Center	6	2	
7	Mavericks	Guard	35	15	
8	Mavericks	Guard	14	8	
9	Mavericks	Forward	16	8	
10	Mavericks	Forward	8	4	
11	Mavericks	Center	3	7	
12					
13					
14					
15					
16					
17					
18					
19					
20					

When applying numeric criteria or comparisons (such as greater than, less than, or not equal to), a critical formatting rule must be followed: the relational operator and the numeric value must be included within the quotation marks. For example, to check for values greater than 5, the criterion is specified as ">5". This is essential for **SUMIFS()** to correctly interpret the threshold condition.

Executing Example 2: Summing Based on Team and Numeric Threshold

To implement this combination of conditions, we structure the formula to reference the Assists column (F) and use the correct relational operator within the criterion argument, while keeping the team criterion from the previous example:

```
=SUMIFS(E2:E10, C2:C10, "Spurs", F2:F10, ">5")
```

The execution of this calculation proceeds through the following logical steps:

The function identifies the target values in **E2:E10** (Points).

It applies the first character filter: checking that **C2:C10** (Team) strictly equals "**Spurs**".

It then applies the secondary numeric filter: ensuring that the corresponding values in **F2:F10** (Assists) satisfy the condition "**>5**".

By reviewing the dataset, we can manually verify the rows that qualify under both filters:

Player 1 (Row 2): Team = Spurs (Matches), Assists = 7 (Matches >5). Points = 31.

Player 2 (Row 3): Team = Spurs (Matches), Assists = 6 (Matches >5). Points = 19.

Player 4 (Row 5): Team = Spurs (Matches), Assists = 8 (Matches >5). Points = 20.

Player 5 (Row 6): Team = Spurs (Matches), Assists = 3 (Fails >5). Points = 12.

The resulting calculation isolates the three rows that satisfied both criteria simultaneously. The output demonstrating the formula's result is shown below:

	A	B	C	D	E	F
1	Team	Position	Points	Assists		
2	Spurs	Guard	31	6		70
3	Spurs	Guard	19	11		
4	Spurs	Forward	24	5		
5	Spurs	Forward	20	6		
6	Spurs	Center	6	2		
7	Mavericks	Guard	35	15		
8	Mavericks	Guard	14	8		
9	Mavericks	Forward	16	8		
10	Mavericks	Forward	8	4		
11	Mavericks	Center	3	7		
12						
13						
14						
15						
16						
17						
18						
19						

The combined total for these three qualifying players (31 + 19 + 20) is **70** points. This successful calculation confirms the power of **SUMIFS()** in handling complex, mixed-criteria requirements.

Advanced Techniques and Best Practices for Reliable SUMIFS Implementation

While the basic implementation of **SUMIFS()** is straightforward, maximizing its utility and ensuring spreadsheet robustness requires adhering to several advanced techniques and best practices.

One powerful feature is the use of **wildcards**. When filtering text criteria, the asterisk (*) acts as a placeholder for any sequence of characters, and the question mark (?) acts as a placeholder for any single character. For instance, using "S*" as a criterion will match "Spurs", "Suns", or any other text string starting with the letter S. Using "???ard" would specifically look for words that are five letters long and end in "ard". Incorporating wildcards allows for flexible matching when the exact text string might vary or when filtering based on partial identifiers or codes, greatly expanding the usefulness of your text-based criteria.

Another crucial best practice involves using **cell references** for criteria instead of hardcoding values. If you type =SUMIFS(..., C2:C10, "Spurs", ...), you must manually update the formula every time the desired team changes. Conversely, if you reference a cell (e.g., A1) containing the value "Spurs", the formula becomes =SUMIFS(..., C2:C10, A1, ...). If the required criterion changes to "Lakers," simply updating cell A1 automatically recalculates the sum, thereby preventing errors and increasing the dynamic nature of your spreadsheet model.

Finally, always ensure that all range arguments--the **sum_range** and every subsequent **criteria_range**--are of **identical dimensions**. If the ranges are mismatched (e.g., summing E2:E10 but checking C2:C15), Google Sheets will typically return an error or produce incorrect results because it cannot align the row-by-row checks necessary for accurate conditional summation. Consistent ranging is non-negotiable for correct formula output.

Conclusion: Elevating Your Google Sheets Data Aggregation

The transition from using the simple SUMIF function to mastering the powerful **SUMIFS()** function marks a significant step forward in your ability to perform sophisticated data aggregation and analysis within Google Sheets. This multi-criteria function provides the precise control necessary to extract meaningful totals from complex datasets where multiple conditions must intersect via the logical AND operator.

By correctly applying the syntax--placing the sum range first and then structuring logical pairs of criteria ranges and criteria--you can handle virtually any conditional summation requirement, whether dealing with multiple character strings, strict numeric thresholds, or dynamic date ranges. Integrating **SUMIFS()** into your analytical toolkit ensures that your reports are not only accurate and reliable but also dynamic and highly specific to the business questions you are trying to answer, saving substantial time compared to manual filtering or array formulas.