

How to Easily Sum Filtered Data in Google Sheets

Authored by
stats writer

December 3, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Sum Filtered Data in Google Sheets*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=104031>

Google Sheets offers powerful analytical capabilities, but calculating sums on data that has been filtered often presents a common challenge for users. When you apply a standard SUM function to a range containing hidden rows, the result unexpectedly includes values from both the visible and the hidden cells, defeating the purpose of the filtering operation. This behavior necessitates the use of specialized functions designed specifically to respect the visibility state of rows within your dataset. Understanding how to correctly implement functions that operate only on visible data is fundamental for accurate financial reporting, statistical analysis, and dynamic dashboard creation within the spreadsheet environment. This guide explores the most effective and straightforward method for summing only the visible, filtered rows using the superior SUBTOTAL function, ensuring your calculations reflect precisely the criteria applied by your filters.

While conditional summing functions like SUMIF, SUMIFS, and DSUM are useful for aggregating data based on specific logical conditions, they do not inherently interact with the visual state of the spreadsheet (i.e., whether a row is hidden by a filter). If you use a filter to visually reduce your dataset to only show records for "Region A," and then apply a standard SUM or even a SUMIF targeting "Region A," the results might still be misleading if the filter criteria themselves are complex or if the underlying goal is to calculate results based purely on what the user currently sees. The key distinction lies in the mechanism: **filtering** hides rows based on applied criteria, whereas functions like SUMIF calculate based on criteria defined within the function argument itself, regardless of row visibility. Therefore, to ensure that calculations dynamically adapt to manual or automated filtering operations, the SUBTOTAL function becomes the indispensable tool.

The Optimal Solution: Utilizing the SUBTOTAL Function

The most robust and universally accepted method for accurately summing a filtered range in Google Sheets is by employing the SUBTOTAL function. Unlike the standard SUM function, SUBTOTAL is specifically designed to perform calculations (such as summing, averaging, counting, finding maximums, etc.) only on cells that are visible within the specified range. When a filter is active, SUBTOTAL intelligently ignores all rows that have been hidden by that filter, providing a true aggregate of the visible data. This capability makes it essential for anyone working with dynamic datasets that frequently undergo filtering for analysis or presentation purposes.

The structure of the SUBTOTAL function is simple yet powerful, requiring only two mandatory arguments. The first argument, known as the **function_code**, dictates which aggregation operation the function will perform. The second argument defines the **range** of cells over which the calculation should be applied. Because we aim to sum the visible rows, we need to select the specific function code associated with the summation operation that ignores hidden rows. This method offers unparalleled efficiency and clarity compared to complex array formulas often suggested for similar tasks in less capable spreadsheet environments.

For the purpose of calculating the sum of visible rows in Google Sheets, the designated **function_code** is **109**. This code corresponds to the **SUM** operation, but with the crucial distinction that it excludes rows hidden by filters. Using this specific function code ensures that when data is filtered--whether by value, color, or condition--the subtotal calculation updates immediately to reflect only the remaining visible entries. The syntax is highly readable and easy to implement, making it the preferred technique for ensuring calculation accuracy in dynamically filtered spreadsheets.

Deconstructing the SUBTOTAL Function Syntax

To perform the summation of a filtered range effectively, the following syntax must be used, which clearly delineates the required arguments for the SUBTOTAL function:

SUBTOTAL(109, A1:A10)

In this structure, the value **109** is the specific numerical code required for the **SUBTOTAL** function to calculate the sum of a filtered range of rows, deliberately excluding any rows that have been hidden by the application of a filter. It is important to note that function codes 1 through 11 (e.g., 9 for SUM) perform the calculation including manually hidden rows, whereas codes 101 through 111 (e.g., 109 for SUM) exclusively ignore rows hidden by the application of the data filtering tool. Always use the three-digit codes (101-111) when your objective is to dynamically update sums based on active filtering rules.

The second parameter, represented here as **A1:A10**, specifies the range of cells containing the numeric values you wish to aggregate. This range should encompass all the data points, including those that might be hidden by the active filter. The **SUBTOTAL** function will then internally analyze the visual state of the rows within this defined range and include only those cell values that correspond to visible rows in its final calculation. This powerful combination of function code and range specification makes the **SUBTOTAL** function indispensable for accurate, dynamic analysis in spreadsheet applications.

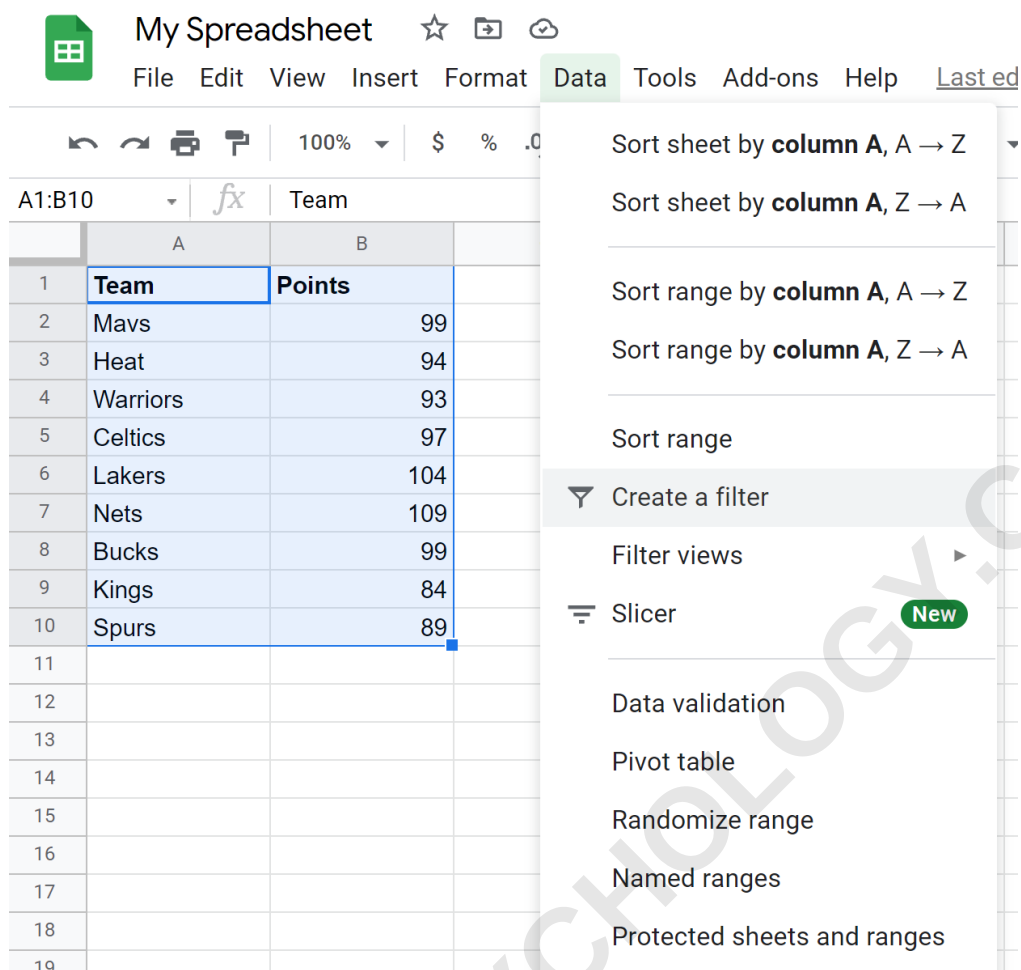
Step-by-Step Example Implementation

Let us walk through a practical example demonstrating how to correctly utilize the **SUBTOTAL(109, range)** function. Suppose we are managing a spreadsheet that tracks performance data for various basketball teams, and we need to calculate the total points scored only for teams meeting certain criteria defined by a filter. The initial unfiltered dataset is presented as follows:

	A	B	C	D
1	Team	Points		
2	Mavs	99		
3	Heat	94		
4	Warriors	93		
5	Celtics	97		
6	Lakers	104		
7	Nets	109		
8	Bucks	99		
9	Kings	84		
10	Spurs	89		
11				
12				
13				
14				
15				
16				
17				

This dataset contains team names and their corresponding points. Our objective is to apply a filter to isolate a subset of teams and then accurately sum the points only for those visible teams.

To begin the process, we must first implement the filtering mechanism on the data. We select the relevant range, in this case, cells **A1:B10**, which includes both the header row and all the data entries. Subsequently, we navigate to the **Data** tab in the Google Sheets menu and select **Create a filter**. This action transforms the headers into filterable columns, indicated by the small filter icon that appears next to the column titles.



The screenshot shows the Google Sheets interface. The 'Data' menu is open, and the 'Create a filter' option is highlighted. The spreadsheet contains a table with the following data:

	A	B
1	Team	Points
2	Mavs	99
3	Heat	94
4	Warriors	93
5	Celtics	97
6	Lakers	104
7	Nets	109
8	Bucks	99
9	Kings	84
10	Spurs	89
11		
12		
13		
14		
15		
16		
17		
18		
19		

Now that the filters are active, we can specify the criteria for hiding certain rows. For this demonstration, we click the Filter icon located at the top of the **Points** column. In the filtering dialogue box that appears, we intentionally uncheck the boxes corresponding to the lower point totals, specifically the values 84, 89, and 93. By unchecking these values, we are instructing the spreadsheet to hide all rows where the Points column contains these specific numerical entries.

The screenshot shows a Google Sheet titled 'My Spreadsheet' with two columns: 'Team' (A) and 'Points' (B). The data is as follows:

Team	Points
Mavs	
Heat	
Warriors	
Celtics	
Lakers	1
Nets	1
Bucks	
Kings	
Spurs	

A filter menu is open over the 'Points' column. The 'Filter by values' option is selected, displaying a list of values: 84, 89, 93, and 94. The value 94 is checked, indicating it is the filter criterion. The 'OK' button is highlighted in green.

Upon clicking **OK**, the spreadsheet updates, and the rows containing the points 84, 89, and 93 are effectively hidden from view. The data is now filtered, presenting only the teams that scored higher totals. At this stage, if an analyst attempts to use the basic `SUM()` function on the entire Points column (e.g., `=SUM(B2:B10)`), the returned value will be the sum of all original values (visible and hidden). This illustrates the inadequacy of the standard `SUM` function when dealing with filtered data, as shown below:

B11		<i>fx</i>	=SUM(B2:B8)		
	A	B	C	D	
1	Team	Points			
2	Mavs	99			
3	Heat	94			
5	Celtics	97			
6	Lakers	104			
7	Nets	109			
8	Bucks	99			
11		695			
12					
13					
14					
15					
16					
17					
18					
19					
20					

Instead, we must correctly utilize the SUBTOTAL function, specifying function code 109 to guarantee that only the currently visible rows contribute to the final sum. The correct formula to place in an aggregation cell (e.g., B11) would be `=SUBTOTAL(109, B2:B10)`. This formula overrides the default summation behavior and delivers the accurate total for the filtered dataset.

B11			=SUBTOTAL(109, B2:B8)		
	A	B	C	D	
1	Team	Points			
2	Mavs	99			
3	Heat	94			
5	Celtics	97			
6	Lakers	104			
7	Nets	109			
8	Bucks	99			
11		602			
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					

The application of the `SUBTOTAL(109, B2:B10)` function yields the precise sum of only the visible rows. In this example, the visible scores are 99, 94, 97, 104, 109, and 99. The resulting sum is $99 + 94 + 97 + 104 + 109 + 99 = 602$. This dynamically calculated value immediately updates whenever the filter criteria are modified, ensuring accurate reporting regardless of the displayed subset of data.

Alternative Methods: Conditional Summing Functions

While `SUBTOTAL(109, ...)` is the definitive method for summing rows hidden by visual filtering, it is important to distinguish it from functions used for conditional summing, such as `SUMIF`, `SUMIFS`, and `DSUM`. These functions are designed to sum values based on criteria defined within the formula itself, rather than based on the visual state of the spreadsheet. If your goal is to sum rows that meet a condition, irrespective of whether a filter is applied, these alternatives are more appropriate.

The `SUMIF` function is used when there is a single criterion that needs to be met before a value is included in the summation. For example, `=SUMIF(A2:A10, "Team X", B2:B10)` would sum the points (in range B) only if the team name (in range A) is "Team X." This function is simpler than `SUBTOTAL` but lacks the dynamic capability to respect visually hidden rows. It operates purely on

the logic defined in its arguments.

When multiple criteria must be satisfied, the `SUMIFS` function is the correct choice. It follows a similar structure but allows for the specification of several criteria ranges and corresponding conditions (e.g., summing points only for teams in "Region A" AND scored over "100" points). While highly effective for complex conditional logic, like `SUMIF`, `SUMIFS` also ignores the visual filtering applied by the user interface; it will still calculate the sum across all rows that meet the formula's criteria, including those hidden by a visual filter.

A more advanced alternative is the `DSUM` function, which belongs to the suite of Database functions in [Google Sheets](#). `DSUM` sums values in a specified column of a database (range) based on a set of criteria defined in a separate criteria range. This approach is powerful for highly structured, database-like analysis where criteria are complex and might change frequently, as the criteria range can be dynamically updated. However, even `DSUM` calculates based on the defined criteria range and does not inherently respect the visual row hiding enforced by the standard data filtering tool, making `SUBTOTAL(109, ...)` the unique solution for filtered row summation.

Best Practices and Advanced Filtering Considerations

When integrating the `SUBTOTAL` function into complex spreadsheets, adherence to certain best practices ensures stability and accuracy. Firstly, always place the `SUBTOTAL` formula outside the data range that is being filtered. If the formula is placed within the range that is being filtered, it can sometimes be hidden, though the calculation itself usually remains correct. Keeping aggregation formulas separate improves readability and accessibility. Secondly, ensure that the reference range (e.g., `A2:A100`) is expansive enough to cover all potential data entries, even as the dataset grows, possibly by utilizing dynamic ranges or named ranges for maximum flexibility.

A critical consideration involves manual row hiding versus filtering. If rows are hidden manually (by right-clicking the row number and selecting "Hide row"), the `SUBTOTAL(109, ...)` function code will still include these manually hidden rows in its calculation. The three-digit function codes (101-111) are designed exclusively to ignore rows hidden by the official **Data > Create a filter** or **Data > Filter views** tool. If you need a function that ignores both manually hidden rows and filtered rows, you would typically need a more complex solution involving an `ARRAYFORMULA` combined with `BYROW` and `ISFILTERED` (though `ISFILTERED` is generally not a direct, easily implemented function in Sheets and often requires scripting). For most standard analytical scenarios, relying solely on the data [filtering](#) feature and `SUBTOTAL(109, ...)` is sufficient and far simpler.

Finally, always verify your results, especially after changing filter criteria. A quick manual check of the visible rows ensures that the function is operating as intended. The power of `SUBTOTAL` lies in its dynamic responsiveness; any modification to the filter instantly recalculates the sum, providing instantaneous, accurate feedback on the filtered subset of data. Mastering this function is key to

generating reliable reports from dynamic spreadsheet data in [Google Sheets](#).

[Google Sheets: How to Sum If Checkbox is Checked](#)

ARABPSYCHOLOGY.COM