How to Easily Calculate Subtotals in Excel VBA

Authored by

stats writer

November 20, 2025

Introduction to the SUBTOTAL Function in VBA

The Visual Basic for Applications (VBA) environment within Excel provides powerful tools for automation and complex data manipulation. Among these tools, the **SUBTOTAL function** stands out as a highly versatile and essential utility. Unlike standard aggregate functions like SUM or AVERAGE, SUBTOTAL offers the unique ability to calculate statistics based exclusively on the **visible cells** within a specified range, making it indispensable when dealing with filtered datasets. This capacity to dynamically adjust calculations based on visibility is what differentiates it from its simpler counterparts, ensuring that your reports and analysis remain accurate even after applying complex data filters.

The primary utility of the SUBTOTAL function lies in its flexibility. It is capable of performing eleven different types of calculations, including calculating sums, counts, averages, and finding minimum or maximum values. Furthermore, its implementation within VBA allows developers to automate sophisticated reporting tasks. By invoking SUBTOTAL through the **WorksheetFunction** object, you gain programmatic control over data aggregation, enabling the creation of custom macros that respond intelligently to dynamic spreadsheet conditions.

Before diving into the practical examples, it is crucial to understand that the function requires two main parameters: the function number (specifying the desired calculation type) and the range reference (specifying the data to be analyzed). Mastery of these parameters is the foundation for leveraging the full power of this tool in your automated Excel solutions. Throughout this guide, we will explore detailed examples illustrating how to define the target Range object and execute the Subtotal calculation effectively.

Understanding the Syntax and Parameters

When utilizing the SUBTOTAL function within a VBA procedure, you must access it via the **WorksheetFunction** object. This object acts as a bridge, enabling VBA to use native Excel worksheet functions. The basic syntax ensures that the result of the calculation is assigned to a specific cell or variable within your macro. The function call itself is straightforward, requiring minimal arguments to perform powerful aggregation.

The core syntax for calling SUBTOTAL in VBA is demonstrated below. This example illustrates how to calculate the sum of values within a defined data range and place the resulting aggregate statistic into a designated output cell. Note the use of the numerical code '9' to specify the desired function type, which corresponds to the standard SUM operation.

```
Sub FindSubtotal()
Range("A16") = WorksheetFunction.Subtotal(9, Range("B2:B11"))
End Sub
```

In the code snippet provided, the macro first targets the output location, which is cell **A16**. It then calls the WorksheetFunction.Subtotal method. The first argument, **9**, instructs the function to perform a summation. The second argument, **Range("B2:B11")**, defines the dataset to be analyzed. Crucially, if any rows within the B2:B11 range are hidden--either manually or through the application of an AutoFilter--the calculation will only include the values from the rows that remain visible, thereby producing an accurate aggregate for the filtered view.

The Crucial Role of Function Arguments (Function_num)

The true versatility of the Subtotal method is unlocked by its first argument, referred to as `Function_num`. This numerical input determines precisely which aggregation calculation the function will execute. There are eleven core functions available, each corresponding to a number between 1 and 11. These numbers represent common statistical operations crucial for data analysis.

Understanding the numeric codes is fundamental for effective use of the function. For example, using the number 1 calculates the average, 2 counts non-blank cells, and 9 computes the sum. Furthermore, the SUBTOTAL function offers a secondary set of codes (101 through 111). When using codes 1 through 11, the function ignores rows hidden by filtering AND rows hidden manually. However, using codes 101 through 111 (e.g., 109 for SUM) causes the function to ignore only rows hidden by filtering, still including rows hidden manually. This distinction offers advanced control over how hidden data is treated.

For the purpose of standard data aggregation on filtered datasets, we typically focus on the standard codes (1-11). The following list details the aggregation methods correlated with their corresponding numerical arguments used in the **Subtotal** method within VBA:

1: AVERAGE (Calculates the arithmetic mean)
2: COUNT (Counts numerical values)
3: COUNTA (Counts non-empty cells)
4: MAX (Finds the largest value)
5: MIN (Finds the smallest value)
6: PRODUCT (Multiplies all numerical values)
7: STDEV (Estimates standard deviation based on a sample)
8: STDEVP (Calculates standard deviation based on the entire population)
9: SUM (Adds numerical values)
10: VAR (Estimates variance based on a sample)
11: VARP (Calculates variance based on the entire population)

Practical Example: Summing Visible Cells

To solidify the understanding of the SUBTOTAL function, let us walk through a practical scenario involving a typical dataset in Excel. Suppose we are managing a spreadsheet containing statistics for various athletes, specifically basketball players. This dataset includes player names, their assigned teams (A, B, C, etc.), and their corresponding point totals. Our goal is to calculate the total points scored, but only for players belonging to specific teams after applying a filter.

Consider the following initial dataset structure. This table represents the raw data before any filtering is applied. It is important to note the cell references, as they will be critical when defining the Range object in our VBA code.

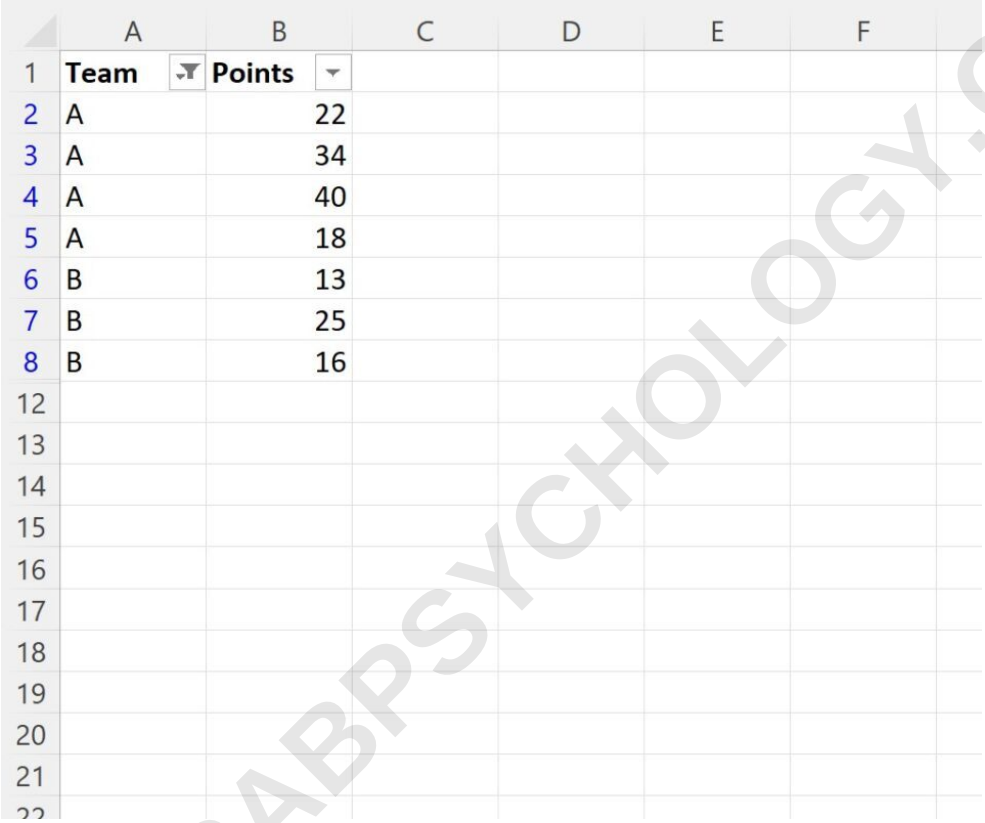
	A	B	C	D	E	F
1	**Team**	**Points**				
2	A	22				
3	A	34				
4	A	40				
5	A	18				
6	B	13				
7	B	25				
8	B	16				
9	C	41				
10	C	11				
11	C	26				
12						
13						
14						
15						
16						
17						
18						

The subsequent steps involve manipulating this data using Excel's native filtering capabilities. Once the filter is applied, we will execute a VBA macro utilizing **WorksheetFunction.Subtotal** to ensure that the calculation reflects only the aggregated scores of the visible players, disregarding those who have been hidden by the filter criteria. This setup effectively demonstrates the power and purpose of using SUBTOTAL over standard aggregation methods like `WorksheetFunction.Sum`.

Applying Filters and Executing the Macro

Following our scenario, we decide to analyze only the performance of players from Team A and Team C. We apply a filter to the 'Team' column, hiding all rows where the team designation is neither A nor C. The resulting visible data subset is significantly smaller, focusing the analysis on the required criteria.

The visual representation of the filtered data clearly shows that only rows corresponding to Team A and Team C remain visible:



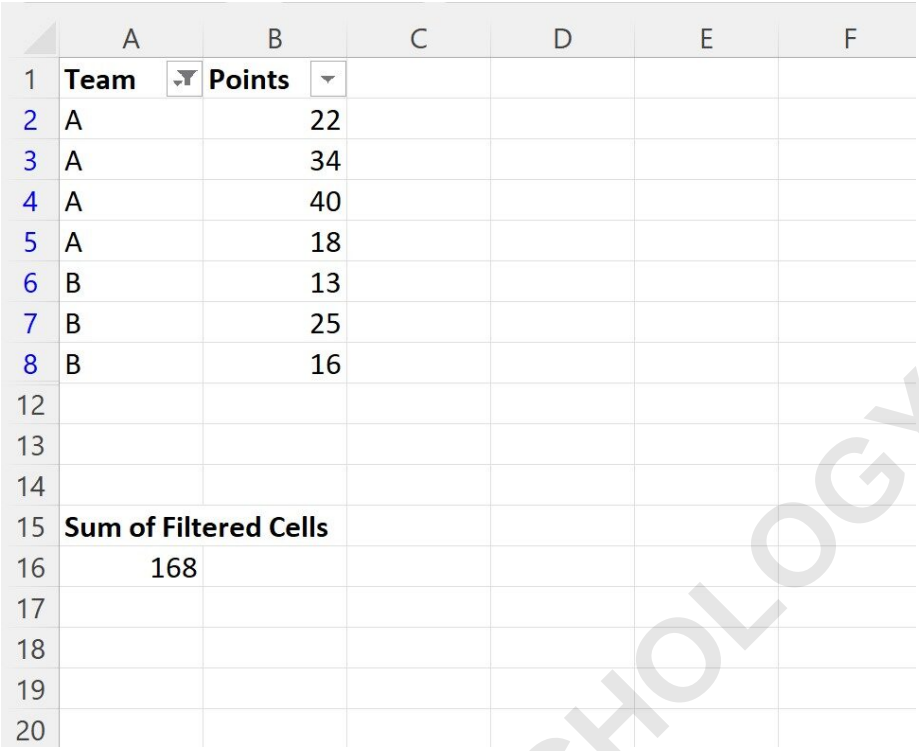
	A	B	C	D	E	F
1	**Team**	**Points**				
2	A	22				
3	A	34				
4	A	40				
5	A	18				
6	B	13				
7	B	25				
8	B	16				
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						

Now, we construct a simple macro designed to perform the summation of the 'Points' column (B2:B11) for these visible cells. We will use the function argument **9** to specify the SUM operation and direct the output to cell **A16**. This approach encapsulates the logic necessary to obtain the dynamically calculated subtotal based on the current filter configuration of the worksheet.

```
Sub FindSubtotal()
Range("A16") = WorksheetFunction.Subtotal(9, Range("B2:B11"))
End Sub
```

Upon execution of the `FindSubtotal` macro, the system calculates the sum of the visible cells

within the specified range. The result is placed directly into cell **A16**, providing an immediate and accurate aggregate score for the filtered teams. The output image below verifies the successful execution of the code and the final calculated subtotal.



	A	B	C	D	E	F
1	**Team**	**Points**				
2	A	22				
3	A	34				
4	A	40				
5	A	18				
6	B	13				
7	B	25				
8	B	16				
12						
13						
14						
15	**Sum of Filtered Cells**					
16	168					
17						
18						
19						
20						

As observed in the resulting spreadsheet, cell **A16** contains the value **168**. This number represents the sum of points (30 + 25 + 18 + 20 + 35 + 40) corresponding only to the players on Teams A and C, effectively ignoring the scores of the hidden players. This confirms the efficacy of using the **Subtotal** method to handle filtered data programmatically.

Adapting the Function: Calculating Averages

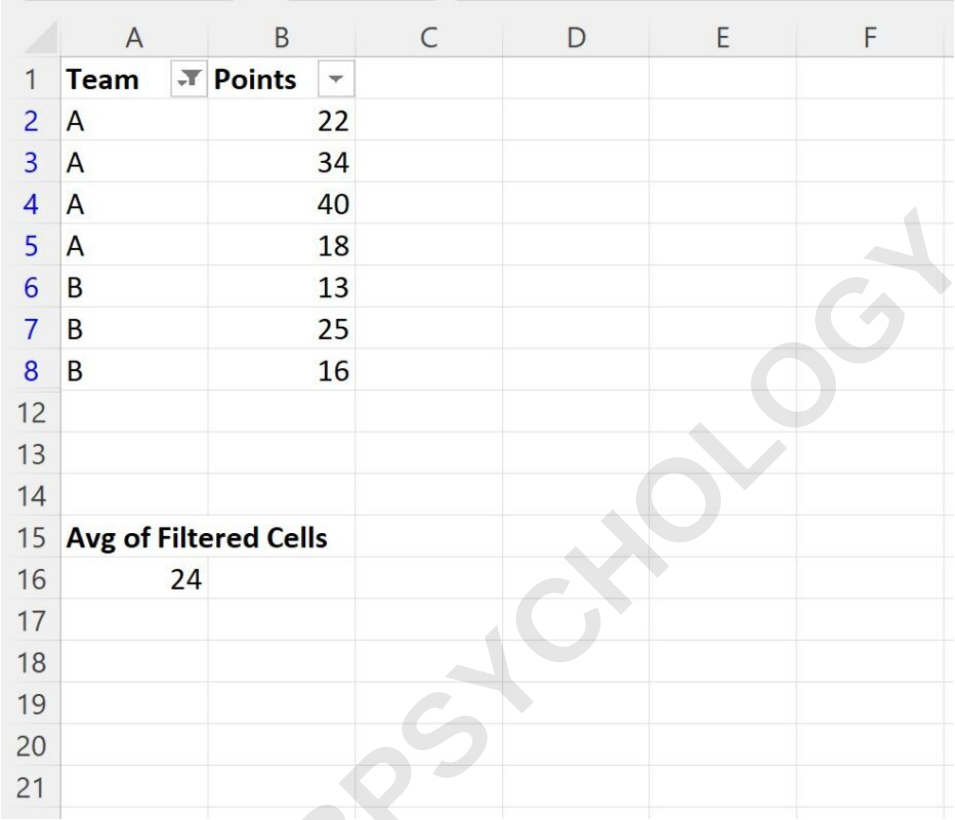
One of the most valuable features of the SUBTOTAL function is the ease with which you can modify the aggregation type without altering the underlying range or filtering logic. By simply changing the `Function_num` argument, we can pivot from calculating the sum to determining other crucial statistics, such as the average, maximum, or count, using the exact same code structure.

To illustrate this flexibility, let us adapt the previous example to calculate the **average** points scored by the visible players (Teams A and C). According to the function code list, the numerical argument for calculating the average is **1**. We replace the argument 9 from the previous sum example with 1 in our VBA code:

Sub FindSubtotal()

```
Range("A16") = WorksheetFunction.Subtotal(1, Range("B2:B11"))
End Sub
```

When this revised macro is executed on the same filtered dataset, the WorksheetFunction object performs the average calculation exclusively on the six visible data points. The result overwrites the previous value in cell **A16**, providing the new statistical measure instantly.



	A	B	C	D	E	F
1	**Team**	**Points**				
2	A	22				
3	A	34				
4	A	40				
5	A	18				
6	B	13				
7	B	25				
8	B	16				
12						
13						
14						
15	**Avg of Filtered Cells**					
16	24					
17						
18						
19						
20						
21						

The resulting value in cell **A16** is **28**. This result clearly indicates that the average score for the visible players in Teams A and C is 28 points. This modification highlights how minor changes in the input arguments can fundamentally change the output, enabling robust and dynamic reporting mechanisms within Excel through VBA automation.

Why SUBTOTAL is Superior to Standard Functions (Summary)

The fundamental advantage of the **SUBTOTAL function** compared to standard functions like `SUM`, `AVERAGE`, or `COUNT` lies entirely in its sensitivity to hidden rows. When standard functions are used on a filtered range, they continue to calculate values based on every cell within the defined range, regardless of whether that cell is visible or hidden. This oversight can lead to misleading or erroneous aggregate statistics when working with filtered data views.

Conversely, SUBTOTAL is purpose-built to recognize and exclude data residing in hidden rows that result from an applied filter. This ensures that any calculation performed--whether summing, counting, or averaging--is always accurate relative to what the user is currently viewing on the spreadsheet. This feature is paramount for creating effective summaries and dashboards where users frequently apply filters to drill down into specific data subsets.

Furthermore, when nesting the SUBTOTAL function within a VBA macro, you gain the power of automation. You can create complex procedures that automatically filter data, calculate multiple subtotals (sum, average, count) using a loop structure and the various function codes, and then output these results into a summary sheet, all without manual intervention. This level of programmatic efficiency makes **Subtotal** a cornerstone tool for any advanced Excel developer.

You can find the complete documentation for the VBA **Subtotal** method and other related functions on the official Microsoft Learn website.

VBA: How to Sum Values in Range