How to Sum Values Between Two Dates in Excel Using a Simple VBA Formula

Authored by

stats writer

November 19, 2025

VBA, or Visual Basic for Applications, is a robust programming language seamlessly integrated into Microsoft Excel. It serves as the definitive tool for automating repetitive tasks, ranging from complex data manipulation to highly specific calculations, such as summing data based on custom date parameters. While standard Excel formulas can handle simple date ranges, utilizing VBA allows users to write a concise macro that efficiently queries a table of dates and corresponding values. This approach provides a powerful and adaptable method for calculating the aggregate sum of values that fall precisely within user-defined start and end dates. VBA offers significant advantages in efficiency, especially when dealing with large datasets or when the summation criteria need to be dynamically updated based on changing inputs, resulting in a far more accurate and professional solution than relying solely on traditional spreadsheet formulas.

Harnessing the WorksheetFunction Object for Conditional Summation

The most effective way to sum values contingent on a date range within VBA is by leveraging the WorksheetFunction object. This critical object exposes almost all of Excel's built-in formulas directly to the VBA environment. Specifically, we rely on the SUMIFS function, which is designed to sum values in a range that meet multiple criteria. When dealing with date ranges, we inherently require two conditions: the date must be greater than or equal to the start date, AND less than or equal to the end date.

Incorporating the WorksheetFunction.SumIfs method into a VBA macro provides immense flexibility. Unlike hardcoding dates into a standard Excel formula, the VBA implementation allows the criteria (the start and end dates) to be pulled dynamically from specific cells. This means the user only needs to update two input cells, and the macro will instantly recalculate the result upon execution, dramatically improving the user experience and reducing the potential for manual calculation errors.

The structured syntax required by VBA ensures that data ranges and criteria ranges are clearly delineated. This clarity is essential for maintaining complex financial or operational models where aggregation based on time periods is a common requirement. The use of comparison operators--specifically the greater than or equal to (`>=`) and less than or equal to (`<=`) operators--is critical in accurately defining the boundary conditions for the date range, ensuring that both the start date and the end date are inclusively counted in the final summation.

Syntax Breakdown of the SumIfs Macro

To effectively calculate the sum of values where the corresponding cells fall between two specified dates, the following concise VBA structure is used. This structure mirrors the arguments required by the standard SUMIFS function within Excel, but it is encapsulated within a sub-procedure for execution.

The general syntax for implementing the conditional sum within VBA is demonstrated below. This code block should be placed within a standard module in the VBA editor (accessible via Alt + F11).

You can use the following syntax in VBA to calculate the sum if cells are between two dates:

```
Sub SumifBetweenDates()
Range("E3") = WorksheetFunction.SumIfs(Range("B2:B9"), Range("A2:A9"), ">=" & , _
Range("A2:A9"), "<=" & )
End Sub
```

This particular configuration is designed to perform a highly targeted calculation. The operation sums the numerical values located in the range **B2:B9** (the sum range), but only applies this summation where the date criteria, found in the corresponding range **A2:A9** (the criteria range), falls precisely between the user-defined start date in cell **E1** and the end date in cell **E2**. Once the calculation is complete, the resulting aggregate sum is then written directly back to cell **E3**, providing an instantaneous output for the user.

Dissecting the WorksheetFunction.SumIfs Arguments

Understanding the precise role of each argument within the WorksheetFunction.SumIfs method is crucial for customization and troubleshooting. The method follows a strict argument order, requiring the summation range first, followed by pairs of criteria ranges and their corresponding criteria.

Range("B2:B9") (Sum Range): This is the column containing the numerical data you wish to add up. In this example, these are the values that will be conditionally summed if they meet the date requirements.

Range("A2:A9") (Criteria Range 1): This range holds the dates that will be tested against the first condition (the start date).

">=" & (Criteria 1): This is the first logical test. It checks if the date in the A2:A9 range is greater than or equal to the value found in cell E1 (our start date). The ampersand (`&`) concatenates the string comparison operator (`">=`) with the actual date value retrieved from cell E1.

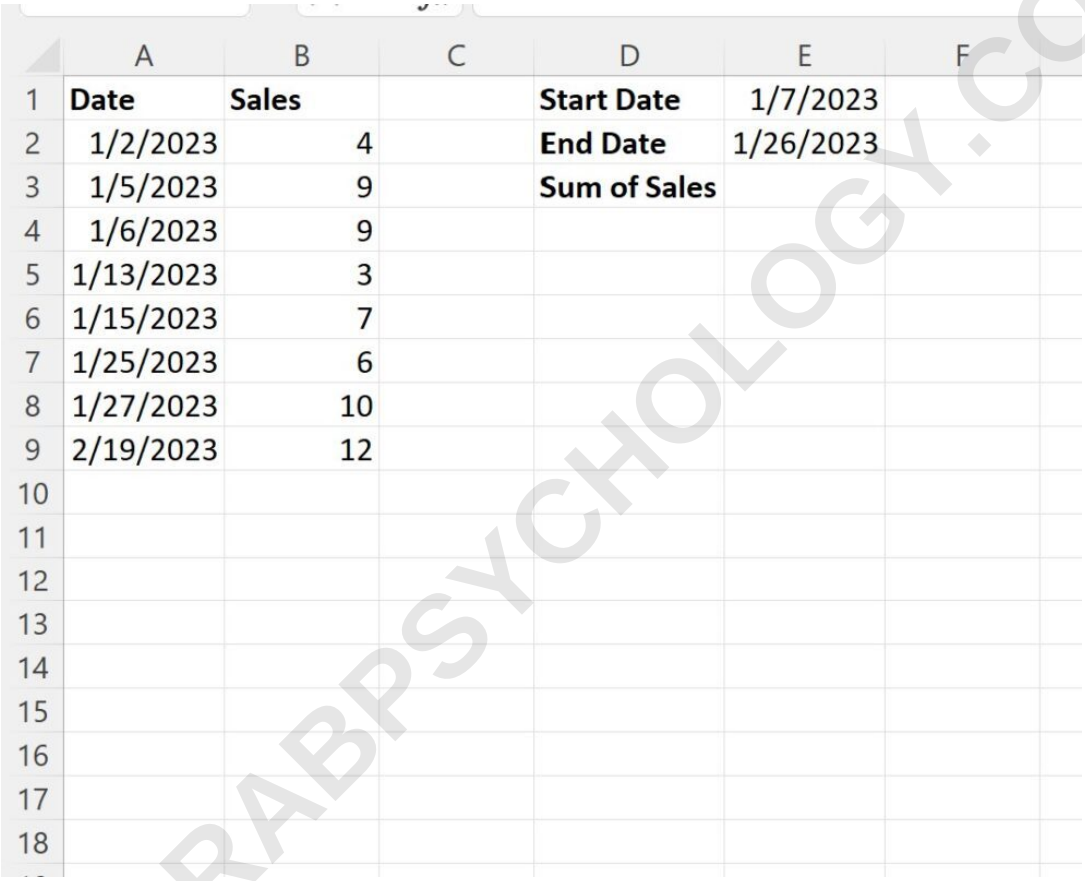
Range("A2:A9") (Criteria Range 2): This is the same date range, used again for the second condition (the end date). SUMIFS requires this range to be repeated for each criterion.

"<=" & (Criteria 2): This is the second logical test. It verifies if the date is less than or equal to the value found in cell E2 (our end date). The combination of these two criteria ensures the date is strictly within the desired time window (inclusive of the start and end dates).

Example: How to Sum If Between Two Dates in VBA

To solidify the understanding of this powerful VBA method, let us walk through a practical business scenario. Imagine a retail store tracking daily product sales. The goal is to quickly calculate the total sales for a specific period without manually filtering the data or altering complex formulas.

Suppose we have the following dataset, meticulously logging the total products sold on various dates. This structure is typical for time-series data analysis where dates are in Column A and corresponding values (Sales) are in Column B:



	A	B	C	D	E	F
1	**Date**	**Sales**		**Start Date**	1/7/2023	
2	1/2/2023	4		**End Date**	1/26/2023	
3	1/5/2023	9		**Sum of Sales**		
4	1/6/2023	9				
5	1/13/2023	3				
6	1/15/2023	7				
7	1/25/2023	6				
8	1/27/2023	10				
9	2/19/2023	12				
10						
11						
12						
13						
14						
15						
16						
17						
18						

Our business requirement dictates that we calculate the sum of sales exclusively for the dates falling between **1/7/2023** (start date) and **1/26/2023** (end date). For this calculation to be dynamic, we designate cell **E1** as the input for the start date and cell **E2** as the input for the end date. The result will then be displayed in cell **E3**.

Implementing and Running the Macro

To execute the summation based on the defined date range, we use the following macro structure. Note that this code remains identical to the structure defined previously, highlighting its

standardized application regardless of the specific dates used. The power lies in its ability to reference the criteria cells (**E1** and **E2**) dynamically.

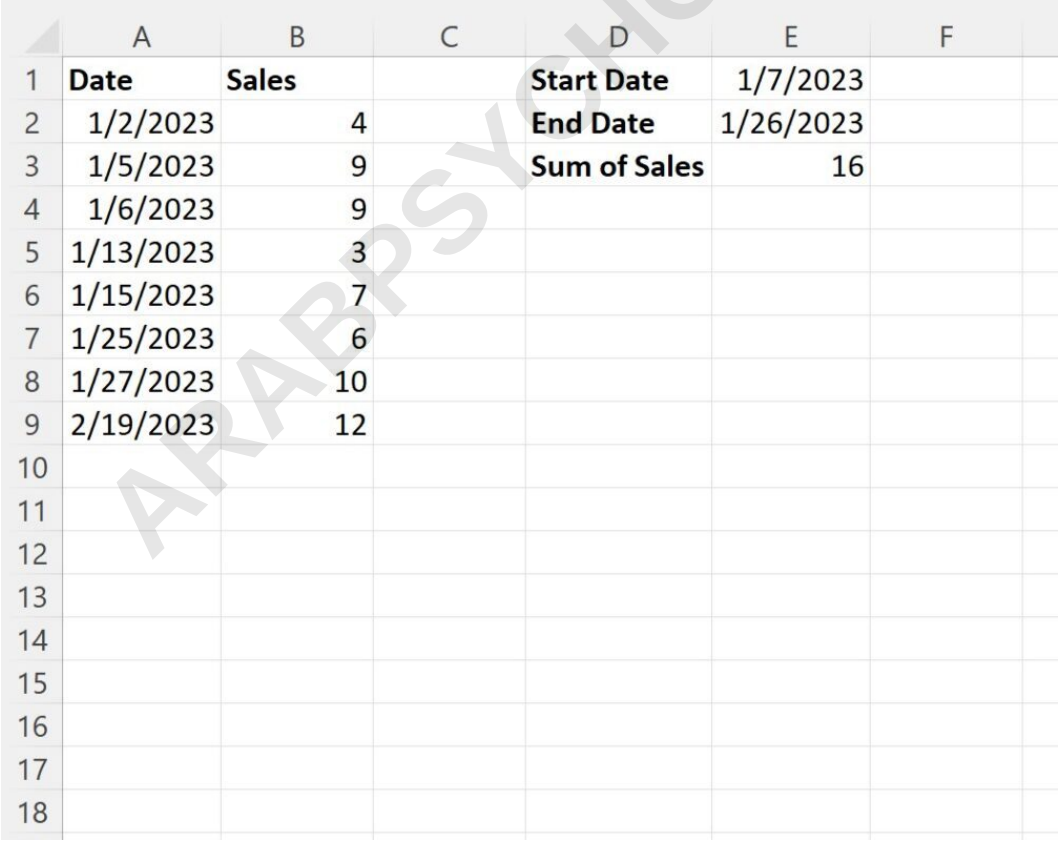
We can create the following macro to achieve this conditional summation:

```
Sub SumifBetweenDates()
Range("E3") = WorksheetFunction.SumIfs(Range("B2:B9"), Range("A2:A9"), ">=" & , _
Range("A2:A9"), "<=" & )
End Sub
```

Upon running this macro (typically by pressing Alt + F8 and selecting "SumifBetweenDates"), VBA instructs Excel to evaluate the dates in Column A against the criteria in cells **E1** and **E2**. Only the corresponding sales figures in Column B that satisfy both conditions will be aggregated.

Verifying the Result and Dynamic Application

When we execute this macro with the start date 1/7/2023 (E1) and end date 1/26/2023 (E2), we receive the following visual output in our spreadsheet:



	A	B	C	D	E	F
1	**Date**	**Sales**		**Start Date**	1/7/2023	
2	1/2/2023	4		**End Date**	1/26/2023	
3	1/5/2023	9		**Sum of Sales**	16	
4	1/6/2023	9				
5	1/13/2023	3				
6	1/15/2023	7				
7	1/25/2023	6				
8	1/27/2023	10				
9	2/19/2023	12				
10						
11						
12						
13						
14						
15						
16						
17						
18						

Observe that cell **E3**, the designated output cell, now contains the resulting value of **16**. This figure

accurately represents the sum of values within the sales column (B2:B9) for which the corresponding date falls inclusively between 1/7/2023 and 1/26/2023.

We can confidently verify the correctness of the VBA calculation by manually reviewing the data entries that meet the criteria:

1/1/2023: Sales 5 (Excluded: Before 1/7/2023)
1/7/2023: Sales 3 (Included: Equals Start Date)
1/15/2023: Sales 7 (Included: Between Dates)
1/26/2023: Sales 6 (Included: Equals End Date)
1/28/2023: Sales 8 (Excluded: After 1/26/2023)
Remaining dates are also outside the specified window.

Manual verification yields the Sum of Sales: 3 + 7 + 6 = **16**. This confirms the accuracy and efficiency of the WorksheetFunction.SumIfs method when executed via VBA. Furthermore, a primary benefit of this approach is its dynamism. If the user changes the start date in cell **E1** and the end date in cell **E2**, running the macro again will instantly recalculate the sum based on the new criteria, eliminating the need to modify the underlying code or complex spreadsheet formulas.

Data Integrity and Formatting Prerequisites

While VBA provides powerful calculation tools, its success hinges entirely on the integrity and correct formatting of the underlying data. It is fundamentally assumed that the values in the date column (Range A2:A9 in this example) are already correctly formatted as dates within Excel. Excel stores dates internally as sequential serial numbers, starting from 1 for January 1, 1900. When comparing dates using the SUMIFS criteria, VBA relies on these numerical values for accurate comparison.

If the data in the date column is stored as text (e.g., resulting from an import process or improper manual entry), the comparison operators (`>=` and `<=`) will fail to recognize them as numerical date values, leading to incorrect or zero results. Therefore, before running any date-based aggregation macro, a strict verification of the date formatting is necessary. Users should ensure that applying the "Short Date" or "Long Date" format to the date column successfully changes the visual appearance, confirming that Excel recognizes them as valid dates.

Advanced Considerations for Dynamic Date Ranges

For more advanced applications, the VBA code can be modified to handle truly dynamic ranges, preventing the code from failing when new rows of data are added. The current example uses fixed ranges (B2:B9, A2:A9), which is acceptable for static datasets but problematic for growing logs.

To enhance robustness, users can define the ranges using VBA logic to find the last occupied row, ensuring that the SUMIFS function always encompasses the entire dataset. This involves using methods like `Cells(Rows.Count, "A").End(xlUp).Row` to determine the boundary of the data. Integrating this dynamic range detection elevates the macro from a simple utility to a highly reliable, production-ready solution for continuous data tracking and reporting in Excel.