How to Sum Values with INDEX MATCH in Excel: A Step-by-Step Guide

Authored by

stats writer

November 19, 2025

The combination of the SUM function with INDEX function and MATCH function is one of the most powerful techniques available to users of Excel for performing flexible lookups and aggregating data. Unlike traditional lookup functions that often restrict the lookup direction or require complex array formulas for summation, this dynamic trio allows you to look up values based on specific criteria and return a sum from a corresponding range or column. This capability is absolutely essential when dealing with large, complex datasets where you need to quickly aggregate numerical values determined by variable text headers or row labels.

Mastering the synergy between INDEX and MATCH provides far greater flexibility than conventional functions like VLOOKUP or HLOOKUP, particularly because INDEX can return an entire range reference rather than just a single cell value. When this range reference is passed into the SUM function, it enables us to sum all values within the dynamically identified column or row, making it an incredibly useful tool for robust reporting and analysis, especially when working with financial data or sales figures that frequently change structure.

Understanding the Core Components

Before diving into the formulas, it is critical to understand the role of each component. The foundation of this method lies in the INDEX function, which returns a value or the reference to a value from within a table or range. Its general syntax is `INDEX(array, row_num, column_num)`. When we omit either the `row_num` or `column_num` by using a zero (`0`), INDEX returns an entire row or column range reference. This ability to return a range, rather than just a single cell value, is what makes it compatible with functions that operate on ranges, such as SUM or AVERAGE.

The MATCH function handles the dynamic search aspect. It searches for a specified item in a range of cells and returns the relative position of that item within the range. For example, if you search for "Apples" in a header row containing "Month, Apples, Oranges," and "Apples" is in the second position, MATCH returns 2. This returned position number is then fed directly into the `row_num` or `column_num` argument of the INDEX function, allowing the lookup to be based on dynamic text criteria rather than fixed column numbers. The final component, SUM, simply takes the resulting range reference generated by INDEX and aggregates all the numerical values within it, providing the desired total.

Method 1: Summing Based on Dynamic Column Value

The first and most straightforward application of this powerful combination is dynamically summing all values within a column identified by a header criteria. This is particularly useful when the column order in your data table might change, or when you need to select which column to total based on user input located in a separate cell. The structure relies on using the MATCH function to identify the position of the header text and passing that position directly to the column argument of the

INDEX function, while setting the row argument to zero.

The structure for this method is shown below, emphasizing how the INDEX function is instructed to return an entire column range by specifying `0` for the row number:

=SUM(INDEX(A2:D6, 0, MATCH(F2,A1:D1,0)))

In this formula, the criteria specified in cell **F2** is looked up across the header range **A1:D1**. The resulting column number is then passed to INDEX, which uses this to dynamically select the correct column from the data range **A2:D6**. Since the row argument is `0`, INDEX returns the entire column reference (e.g., C2:C6), which is then processed by the outer SUM function to calculate the total. This technique is clean, efficient, and avoids the volatility issues sometimes associated with array-based calculations using SUMPRODUCT for single column sums.

Method 2: Summing Based on Row and Column Values (Two-Dimensional Criteria)

When you need to sum values that meet not just a column criterion, but also a row criterion (e.g., finding the total sales for "Bananas" but only for the month "January"), we introduce the SUMIF function. In this advanced setup, the INDEX/MATCH combination is used exclusively to dynamically determine the **Sum Range** for SUMIF, while the SUMIF function itself handles the row-level conditional summing.

This method is significantly more efficient than attempting a complex nested array formula for two-dimensional lookups when one criterion is fixed (like a month or category label) and the other is dynamic (like a header name). The formula leverages MATCH to find the correct column position, INDEX to return the correct column range, and SUMIF to filter that range based on the row criteria.

Here is the structure for performing a conditional sum based on both row and column inputs:

=SUMIF(B2:B9, G2, INDEX(C2:E9,0,MATCH(H2,C1:E1,0)))

In this complex structure, the SUMIF function first checks the criteria range **B2:B9** (which contains the row labels like months or locations) against the criteria in cell **G2** (the desired row label). The crucial part is the final argument, the `sum_range`, which is dynamically provided by `INDEX(C2:E9, 0, MATCH(H2, C1:E1, 0))`. This nested formula identifies the correct column (based on the value in **H2**) within the data matrix **C2:E9** and returns that column as the range to be summed, allowing the SUMIF function to apply the row-level filter to the correct data column.

The following examples illustrate how to implement each method successfully in a real-world data

scenario using Excel.

Example 1: Summing Sales Based Solely on Column Header

For our first practical demonstration, assume we are managing a dataset showing the total sales of several different types of fruits across various locations, represented by rows. Our objective is to calculate the total sales for a specific fruit, regardless of the location. This requires us to dynamically identify the column associated with that fruit name and sum all numerical values found within it. This demonstrates the power of the basic `SUM(INDEX(MATCH()))` structure to perform vertical aggregation based on horizontal criteria.

Consider the sample dataset displayed below, which tracks monthly sales figures. The data range includes the locations in Column A, and sales figures for different fruits in Columns B, C, and D. Our goal is to extract the total sales for "Bananas" using a dynamic lookup formula, ensuring that if the "Bananas" column were moved, the formula would still accurately identify its location and return the correct total.

	A	B	C	D	E	F
1	**Month**	**Apples**	**Bananas**	**Oranges**		
2	January	10	5	13		
3	February	4	5	14		
4	March	8	4	5		
5	April	7	5	8		
6	May	2	7	8		
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						

To achieve our goal of summing all sales for the column where the fruit is identified as **Bananas**, we will structure the formula to first locate the "Bananas" header within the range A1:D1, and then

instruct the INDEX function to return the corresponding data column from A2:D6. We will place our lookup criteria ("Bananas") in cell F2 and the resulting formula in cell G2.

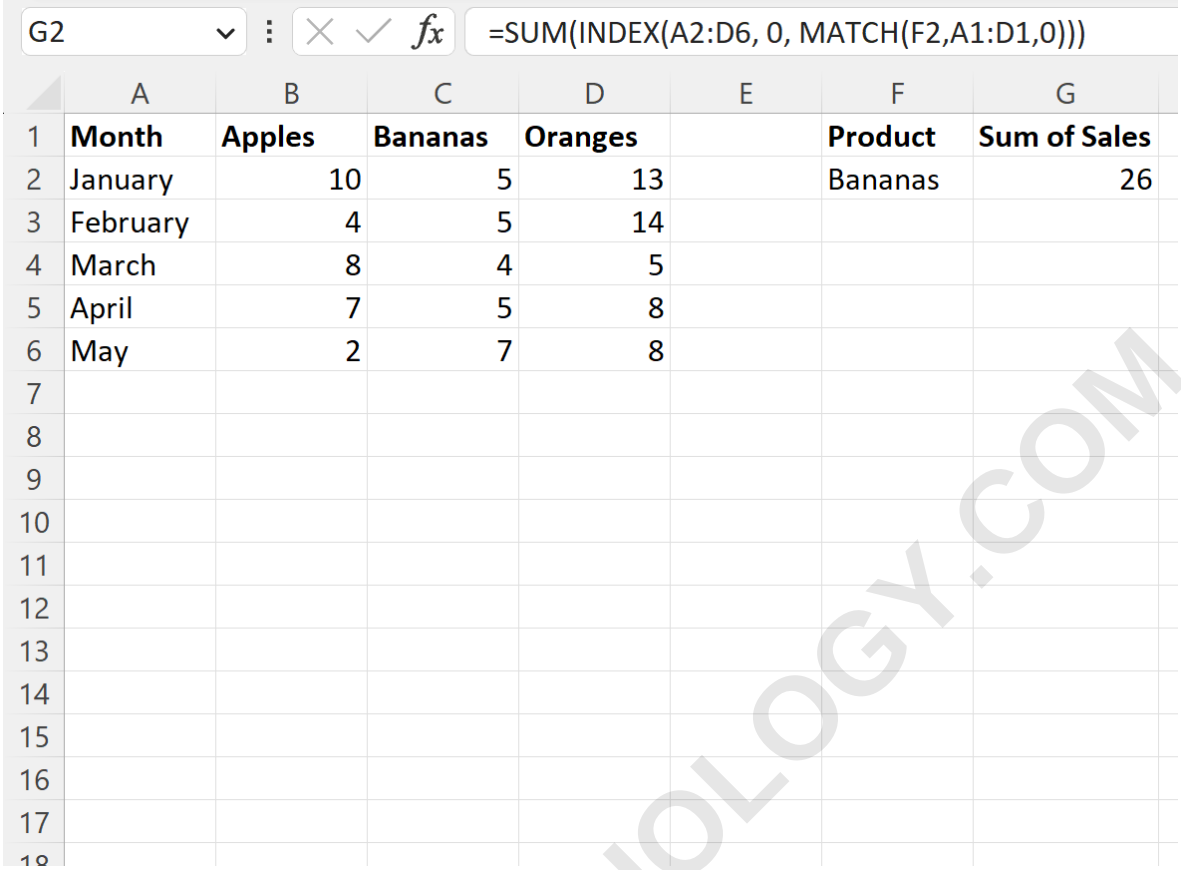
Executing the Column Value Lookup

To begin the calculation, we enter the required structure into cell **G2**. We ensure the formula references the entire data set **A2:D6** for the array argument of INDEX, use `0` for the row number to indicate "return the whole column," and use the MATCH function to calculate the column number dynamically based on the input cell **F2** against the header row **A1:D1**. The precise formula is:

=SUM(INDEX(A2:D6, 0, MATCH(F2,A1:D1,0)))

When Excel evaluates this formula, it first processes the MATCH function: it finds "Bananas" (F2) in the range A1:D1, determining its relative position is the third column (since the range starts at A1). This value (3) is then passed to the INDEX function. INDEX then looks at the range A2:D6 and returns the entire third column of data (C2:C6). Finally, the SUM function aggregates these values.

The result of applying this formula is clearly illustrated in the following screenshot, showing the input criteria and the calculated output:



	A	B	C	D	E	F	G
1	**Month**	**Apples**	**Bananas**	**Oranges**		**Product**	**Sum of Sales**
2	January	10	5	13		Bananas	26
3	February	4	5	14			
4	March	8	4	5			
5	April	7	5	8			
6	May	2	7	8			
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							

The calculated result returned by the formula is **26**. We can manually verify this total by looking at the sales figures for Bananas (Column C): 5 + 5 + 4 + 5 + 7 = **26**. This demonstrates the precise and dynamic calculation capability of `SUM(INDEX(MATCH()))` for aggregating values based on flexible column identification.

Example 2: Summing Sales Based on Row (Month) and Column (Fruit) Criteria

In scenarios requiring conditional summation across two dimensions--where both the row and the column must meet specific criteria--we utilize the enhanced structure combining SUMIF with INDEX/MATCH. For this example, imagine we have sales data segmented not only by fruit but also by the month in which the sales occurred. We want to find the total sales for a specific fruit, but only during a specified month. This requires filtering the data vertically by month and dynamically identifying the correct column horizontally by fruit name.

The dataset for this example, shown below, includes a new column (B) detailing the month of the transaction. We aim to calculate the sum of all sales where the fruit is **Bananas** AND the transaction occurred in **January**. This requires the formula to successfully navigate the two-

dimensional nature of the data table, ensuring accuracy and flexibility even if the fruit columns were rearranged.

	A	B	C	D	E	F
1	**Store**	**Month**	**Apples**	**Bananas**	**Oranges**	
2	A	January	10	5	13	
3	A	February	4	5	14	
4	A	March	8	4	5	
5	A	April	7	5	8	
6	B	January	2	7	8	
7	B	February	4	2	8	
8	B	March	9	4	7	
9	B	April	7	4	2	
10						
11						
12						
13						
14						
15						
16						
17						
18						

Our objective is to calculate the sum of sales where the row value (Month) is equal to **January** and the column value (Fruit) is equal to **Bananas**. We will place the Month criteria ("January") in cell G2 and the Fruit criteria ("Bananas") in cell H2. The calculated result will be placed in cell I2. This configuration allows users to easily change both the row and column criteria without altering the underlying formula structure, providing high analytical versatility.

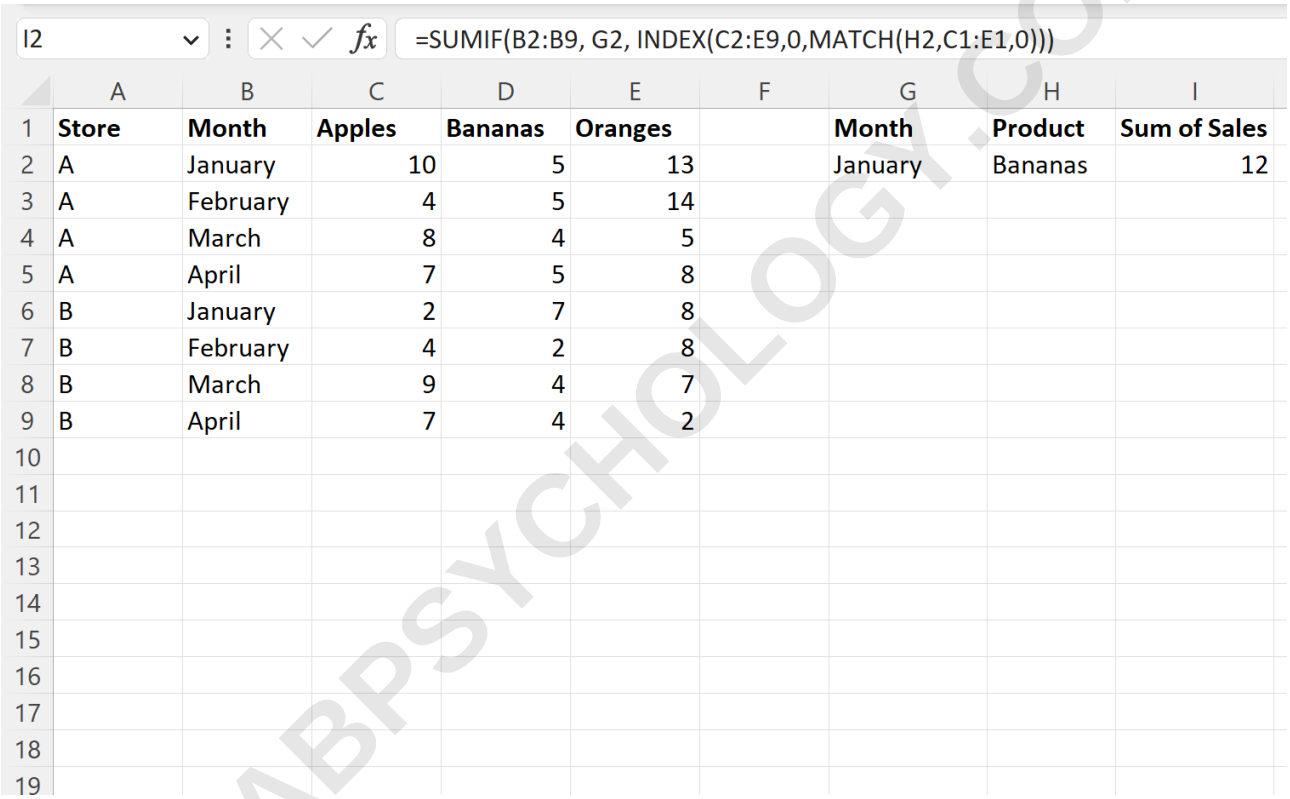
Executing the Row and Column Criteria Lookup

To perform this two-dimensional conditional summation, we input the specialized formula into cell **I2**. The formula uses the SUMIF function to control the row criteria (Month). The range **B2:B9** is the criteria range, and **G2** is the criteria ("January"). The final argument, the sum range, is generated dynamically by the nested INDEX/MATCH:

=SUMIF(B2:B9, G2, INDEX(C2:E9,0,MATCH(H2,C1:E1,0)))

The inner MATCH function first determines the position of "Bananas" (H2) within the headers (C1:E1). Assuming "Bananas" is the second item, the position 2 is passed to INDEX. INDEX then extracts the entire second column of the data range C2:E9 (which is column D). This column range D2:D9 is now provided to SUMIF as the target sum range. SUMIF then checks B2:B9 for all occurrences of "January" (G2) and aggregates the corresponding values from the dynamically provided column D2:D9.

The outcome of this calculation is captured in the following screenshot, illustrating the sophisticated interaction between the functions:



	A	B	C	D	E	F	G	H	I
1	**Store**	**Month**	**Apples**	**Bananas**	**Oranges**		**Month**	**Product**	**Sum of Sales**
2	A	January	10	5	13		January	Bananas	12
3	A	February	4	5	14				
4	A	March	8	4	5				
5	A	April	7	5	8				
6	B	January	2	7	8				
7	B	February	4	2	8				
8	B	March	9	4	7				
9	B	April	7	4	2				
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									

The calculation successfully returns a value of **12**. We can confirm this result by manually checking the dataset for entries where the Month is "January" (Column B) and the fruit is "Bananas" (Column D). These entries are found in rows 2 and 4, showing values of 5 and 7, respectively. The sum of Sales for Bananas in January is therefore 5 + 7 = **12**. This match validates the successful implementation of the `SUMIF(INDEX(MATCH()))` method for complex, two-criteria summation in Excel.

Conclusion: The Efficiency of Dynamic Summation

The use of SUM combined with INDEX/MATCH, optionally enhanced by SUMIF, provides an exceptionally reliable and adaptable toolset for data aggregation in Excel. This method surpasses

the limitations of traditional lookup functions by enabling dynamic column identification, making your workbooks more resilient to structural changes in the source data.

Whether you are summing an entire column based on a dynamic header (Method 1) or adding an additional row criterion using SUMIF (Method 2), these techniques ensure that your reports remain accurate and maintain integrity even as datasets evolve. By learning to deploy these combinations effectively, users can transition from simple static analysis to highly dynamic and powerful reporting capabilities.