Understanding Excel: How to Sum Cells Containing Both Text and Numbers

Authored by

Mohammed looti

October 28, 2025

When working with data in Excel, it is common to encounter cells that contain both numerical values and descriptive text. This situation often arises from poor data entry practices, or when importing data from external sources that automatically append units or descriptions (e.g., "45 items," "120 units"). Standard functions like the SUM function cannot process these cells directly because Excel interprets the mixed content as a text string, not a number. To accurately calculate totals from such mixed data, we must employ a powerful combination of string manipulation and numerical conversion techniques.

The primary solution involves using the SUBSTITUTE function to surgically remove the unwanted text, followed by an arithmetic operation to coerce the resulting text string back into a usable numerical format. This process allows the subsequent SUM function to aggregate the cleaned values successfully. The fundamental structure for solving this pervasive data challenge is demonstrated in the formula below, which is designed to handle ranges containing uniform text identifiers.

=SUM(SUBSTITUTE(B2:B8, "some_text", "")+0)

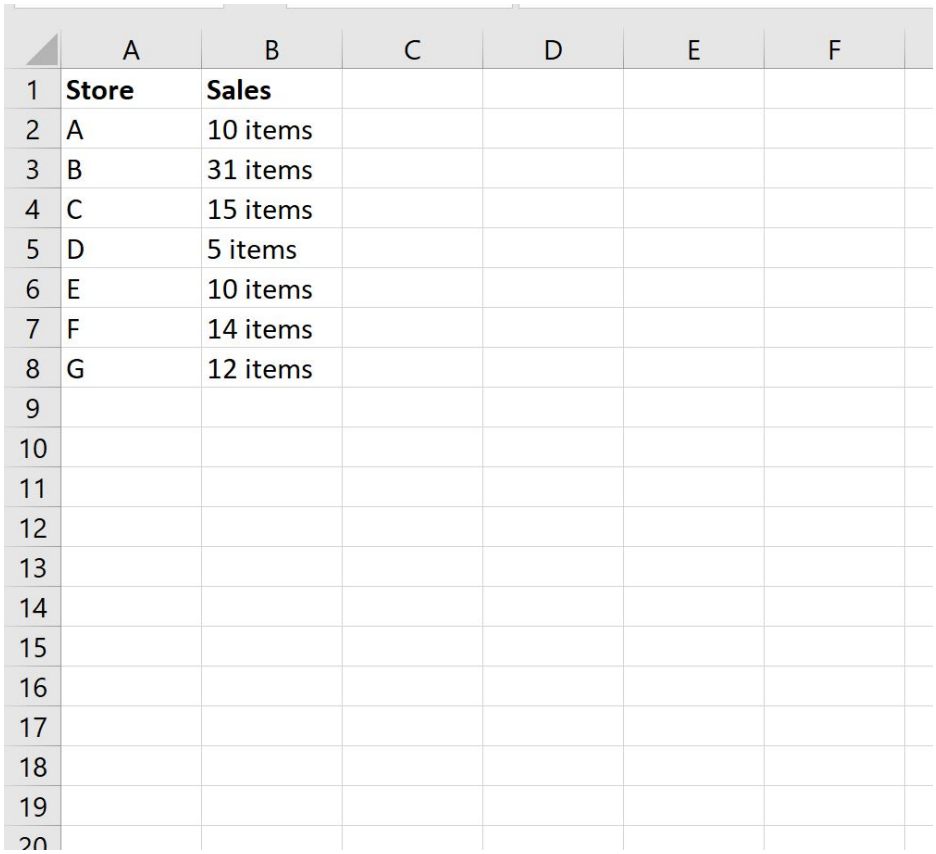
This particular formula operates by systematically iterating through the specified range, **B2:B8**. The inner SUBSTITUTE function is responsible for identifying and replacing the text string "some_text" with an empty string (""), effectively isolating the numerical component of the cell. The critical addition of `+0` performs an implicit type coercion, forcing Excel to convert the resulting array of text numbers into an array of actual numerical values, which the outer SUM function can then reliably calculate.

To fully grasp the mechanics of this powerful technique, the following examples illustrate step-by-step implementations, addressing both scenarios where the descriptive text is consistent across the range and where multiple different text strings must be removed simultaneously.

Example 1: Calculating Sums with Uniform Text Strings

A common scenario involves a dataset where every entry includes the same unit descriptor. For instance, consider a sales log where the recorded volume of sales at seven different retail locations is consistently followed by the word "items." Because the text is uniform, we only require a single substitution operation to clean the data before summation. This approach ensures efficiency and accuracy when dealing with high-volume, standardized data entry.

Suppose we have the following dataset that shows the total number of sales at seven different stores, as visually represented below. Notice that the numerical value in column B is inseparable from the text " items."

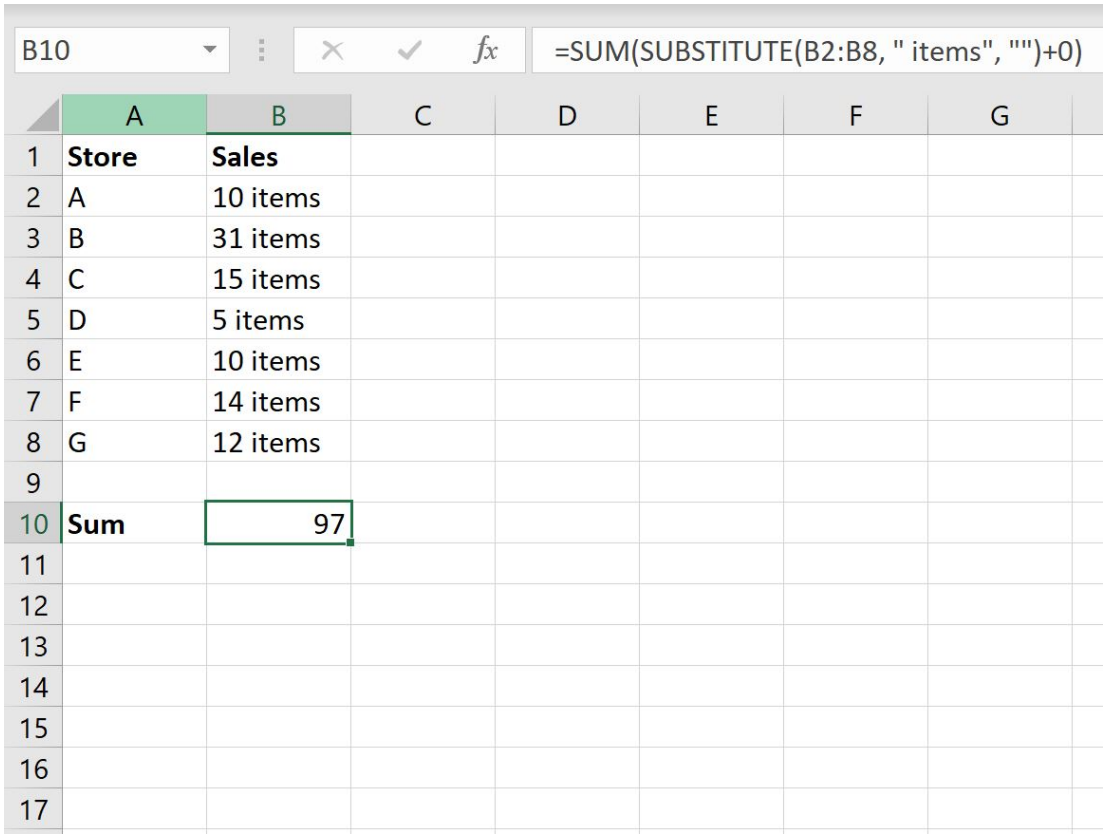


	A	B	C	D	E	F
1	**Store**	**Sales**				
2	A	10 items				
3	B	31 items				
4	C	15 items				
5	D	5 items				
6	E	10 items				
7	F	14 items				
8	G	12 items				
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

To correctly calculate the sum of sales from this range (B2 through B8), we must instruct Excel to ignore the recurring text string. We can achieve this by typing the following robust formula directly into cell **B10**, or any other dedicated calculation cell. The key is to include the leading space before " items" within the quotes to prevent any partial substitutions that might occur if the space were omitted.

=SUM(SUBSTITUTE(B2:B8, " items", "")+0)

Upon confirming the formula by pressing **Enter** (or Ctrl+Shift+Enter for older versions of Excel that require Array formula entry), the final, consolidated sum of the items will be displayed. The formula effectively processes the range, creating an intermediate array of pure numbers that the SUM function then aggregates, demonstrating the power of array manipulation in spreadsheet software.



As clearly demonstrated in the resulting output, the calculated sum of the items sold across all seven stores is accurately determined to be **97**. This calculation validates the method: the SUBSTITUTE function successfully stripped the extraneous text, and the subsequent arithmetic conversion prepared the resulting values for accurate summation.

Fundamentally, this formula operates through substitution, replacing the specific string " items" with nothing (a blank space). Once the text is removed, the remaining values, which still technically reside as text strings in an array format, are converted into true numerical types via the `+0` operation. This crucial step ensures that the final SUM function receives numerical data it can process, rather than an array of unsummable text values.

Example 2: Handling Multiple, Diverse Text Strings

Data integrity challenges often extend beyond uniform text strings. It is highly probable that a dataset imported from various sources may contain different unit descriptors or miscellaneous text artifacts embedded alongside the desired numbers. When multiple distinct text strings must be eliminated from the same range, we must nest multiple SUBSTITUTE functions within each other. This nesting allows the formula to sequentially scrub the data of all known interfering text elements before the final calculation.

Imagine a scenario similar to Example 1, where we track sales across seven stores, but this time, some stores report sales as " items" while others report them as "things." Although the underlying numerical data represents the same metric (sales volume), the differing text strings complicate straightforward extraction. The following dataset illustrates this common data cleaning challenge:

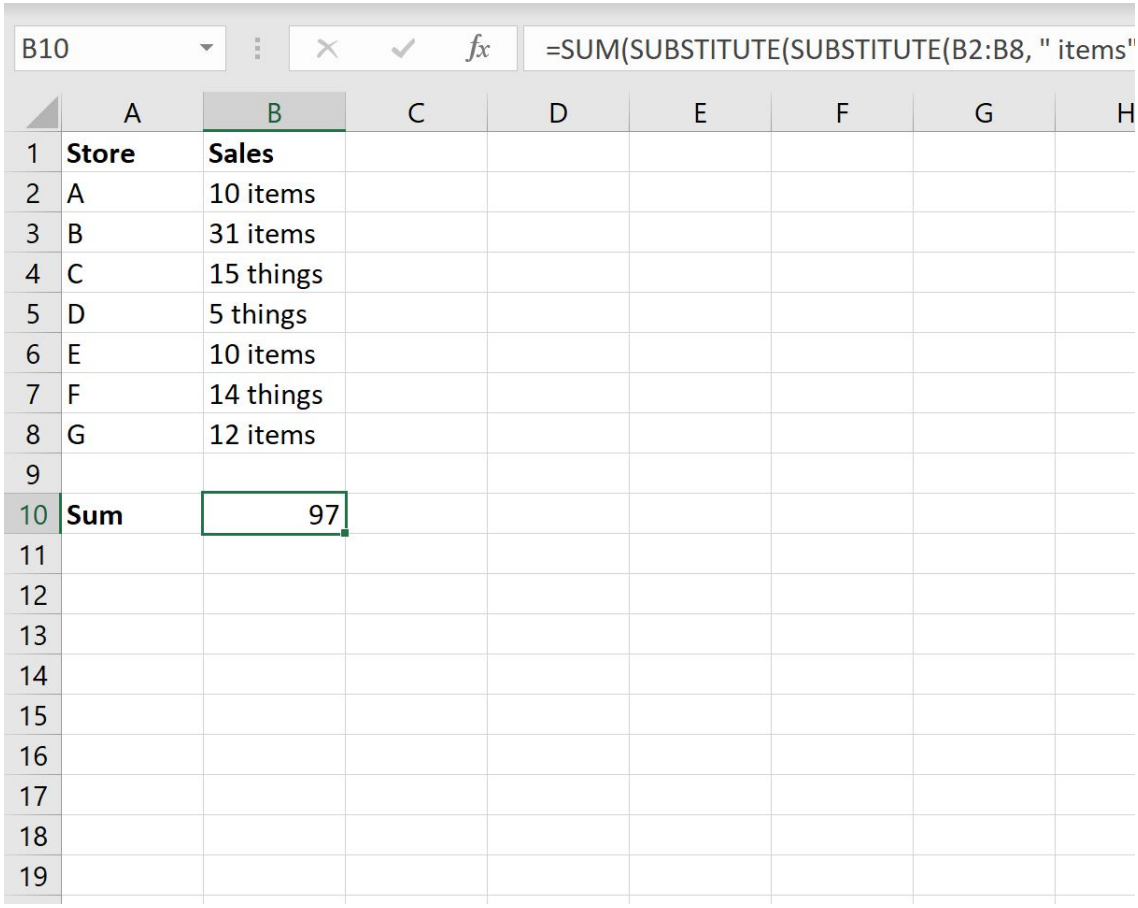
	A	B	C	D	E	F
1	**Store**	**Sales**				
2	A	10 items				
3	B	31 items				
4	C	15 things				
5	D	5 things				
6	E	10 items				
7	F	14 things				
8	G	12 items				
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						

To calculate the accurate total sum of sales from this mixed range, we need a refined formula that addresses both text strings, " items" and "things." The solution is to nest a second SUBSTITUTE function around the first. The innermost function removes the first text type, and the output of that operation feeds directly into the next SUBSTITUTE function, which removes the second text type. We input the following formula into cell **B10**:

=SUM(SUBSTITUTE(SUBSTITUTE(B2:B8, " items", ""), "things", "")+0)

The evaluation of this nested formula proceeds from the inside out. First, Excel processes the range **B2:B8**, removing all occurrences of " items." The resulting intermediate array, which still contains "things," is then passed to the outer SUBSTITUTE function, which removes "things." Once the data is entirely clean of text, the `+0` operation converts the array of clean text strings into usable numerical values, enabling the final SUM calculation.

Once we press **Enter**, the comprehensive sum of the values in column B, after successfully eliminating both text artifacts, will be displayed in the designated calculation cell. This result confirms that nesting the SUBSTITUTE functions is an effective strategy for handling heterogeneous data cleanliness requirements.



Just like in the previous example, the sum of the items sold remains **97**. This demonstrates the consistency and reliability of the nested substitution method, regardless of the variety of text strings present in the column. The key takeaway is that for every unique text string that needs to be removed from the range, an additional, nested SUBSTITUTE function must be added to the formula structure.

Why the "+0" is Crucial: Implicit Type Coercion

While the SUBSTITUTE function successfully cleans the data by removing the unwanted text, its output remains an array of text strings, even if those strings contain only digits. If we were to omit the `+0` (or a similar mathematical operator) and simply wrap the clean substitution array in the SUM function, the result would be zero, or an error, because the SUM function is designed to ignore non-numeric text values within its range.

The addition of a mathematical operation, such as adding zero (`+0`), multiplying by one (`*1`), or using the double unary operator (`--`), triggers a process known as **implicit type coercion**. Excel recognizes that a calculation is being performed, and since it cannot perform arithmetic on text, it automatically attempts to convert the text strings into their numerical equivalents before executing the operation.

For example, when Excel sees the text string "15" and is asked to perform "15" + 0, it first converts "15" into the numerical value 15. Since this arithmetic operation is applied to the entire array generated by the SUBSTITUTE function, the result passed to the outer SUM function is a fully numerical array, ready for aggregation. This subtle yet vital step is what transforms the data cleaning operation into a successful numerical calculation.

Additional Resources

Mastering data manipulation techniques in Excel is essential for anyone dealing with real-world, often messy, datasets. The formula demonstrated here is a cornerstone for cleaning imported or poorly formatted quantitative data. To further enhance your proficiency in data handling and analysis within spreadsheet environments, consider exploring the following related tutorials and concepts:

Detailed explanation of Array formula behavior and execution in various versions of Excel.
Using the SUBSTITUTE and SUM functions for more complex text extraction patterns.
Techniques for handling numbers embedded with currency symbols or punctuation using similar substitution methods.

The following tutorials explain how to perform other common tasks in Excel: