Learn How to Remove Substrings in Google Sheets: A Step-by-Step Guide

Authored by

Mohammed looti

June 2, 2026

Understanding String Manipulation in Spreadsheets

When working with large datasets in tools like Google Sheets, it is extremely common to encounter inconsistencies or unwanted textual elements that require cleaning. One of the most frequent data manipulation tasks involves removing specific segments of text--known as a substring--from various cells. While this might seem complex, Google Sheets provides powerful, built-in functions designed specifically for text transformation.

This comprehensive guide will demonstrate the primary methods for stripping out unwanted substrings, whether you need to target a single phrase or handle multiple different text patterns within your dataset. We rely predominantly on the versatile **SUBSTITUTE** function, which allows for precise control over text replacements. Understanding these techniques is fundamental for effective data cleaning and preparation, ensuring your data is standardized for analysis or integration with other systems.

The key principle utilized in these methods is not deletion, but rather substitution. By replacing the unwanted substring with an empty string (represented by `""`), we effectively remove the target text, leaving the remainder of the cell content intact. This approach is highly reliable and prevents common data errors associated with manual text editing.

The Power of the SUBSTITUTE Function

The core solution for removing substrings in Google Sheets lies with the **SUBSTITUTE** function. Essentially, this function operates by finding a specific text pattern (the substring you wish to remove) within a larger string and replacing it with another string. To achieve the effect of "removal," we simply replace the target substring with an empty string, represented by two double quotes (`""`).

The syntax for the **SUBSTITUTE** function is straightforward, yet incredibly powerful: `=SUBSTITUTE(text_to_search, search_for, replace_with, )`. The first three arguments are mandatory: `text_to_search` refers to the cell containing the original data (e.g., A2); `search_for` is the exact substring you want to eliminate (e.g., "Team"); and `replace_with` is the string that will take its place, which, for removal, is always `""`. The optional fourth argument, `occurrence`, specifies which instance of the substring to replace if it appears multiple times in the same cell. When this argument is omitted, as in our general examples, the function intelligently replaces every instance found.

Below are the foundational formulas we will explore. Note how simple the concept is: we are instructing the spreadsheet to substitute the unwanted text with nothing, thereby achieving the desired removal.

Method 1: Removing a Single Substring from a Cell

=SUBSTITUTE(A2,"this_string","")

Method 2: Removing Multiple Substrings from a Cell (Nesting)

=SUBSTITUTE(SUBSTITUTE(A2,"string1",""),"string2","")

The practical examples that follow will illustrate exactly how to deploy these formulas to cleanse and standardize your data effectively.

Method 1: Efficiently Removing a Single Substring

When your dataset contains a consistent, single piece of text that needs to be removed across all records--such as a redundant label or prefix--Method 1 offers the fastest and most direct approach. This is ideal for standardization purposes, ensuring that all entries conform to a clean structure necessary for further analysis or database migration. We utilize the SUBSTITUTE function once per cell reference.

Consider a scenario where we are analyzing basketball team statistics, but the data source automatically appended the word "Team" to every entry, which complicates sorting and lookup operations. Suppose we have the following dataset that contains information about various basketball teams:

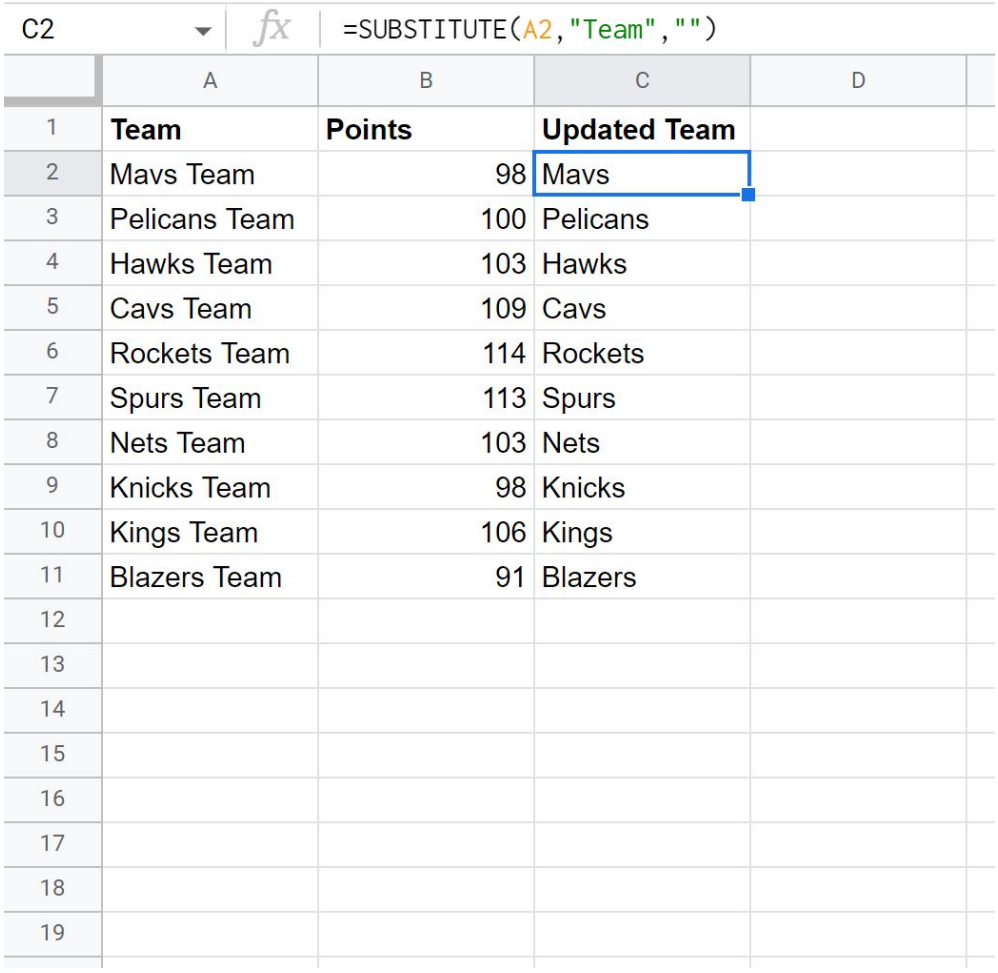
	A	B	C	D
1	**Team**	**Points**		
2	Mavs Team	98		
3	Pelicans Team	100		
4	Hawks Team	103		
5	Cavs Team	109		
6	Rockets Team	114		
7	Spurs Team	113		
8	Nets Team	103		
9	Knicks Team	98		
10	Kings Team	106		
11	Blazers Team	91		
12				
13				
14				
15				
16				
17				
18				

Our objective is to isolate the team name itself by eliminating the redundant word "Team" from every entry in Column A. To achieve this, we apply the single-layer SUBSTITUTE formula. Since we want to remove the text entirely, the replacement argument will be `""`. The formula targets cell A2, which contains the first team name, and specifies "Team" as the text to be removed. It is important to note that the SUBSTITUTE function is case-sensitive; if the substring were "team" (lowercase), the formula would need to reflect that exact casing.

We use the following formula to remove the substring "Team" from the content of cell A2:

=SUBSTITUTE(A2,"Team","")

To apply this transformation across the entire dataset, we input this formula into cell **C2**. Subsequently, we use the fill handle (dragging the bottom-right corner of cell C2) to apply the formula down to the remaining cells in column C. This action dynamically updates the cell reference (A2 changes to A3, A4, and so on) for each row, performing the necessary substitution on all corresponding entries in Column A.



	A	B	C	D
1	**Team**	**Points**	**Updated Team**	
2	Mavs Team	98	Mavs	
3	Pelicans Team	100	Pelicans	
4	Hawks Team	103	Hawks	
5	Cavs Team	109	Cavs	
6	Rockets Team	114	Rockets	
7	Spurs Team	113	Spurs	
8	Nets Team	103	Nets	
9	Knicks Team	98	Knicks	
10	Kings Team	106	Kings	
11	Blazers Team	91	Blazers	
12				
13				
14				
15				
16				
17				
18				
19				

As illustrated by the results in Column C, the substring "Team" has been successfully and cleanly removed from each data point, leaving only the desired team name. This process demonstrates the efficiency of the single SUBSTITUTE function for targeted, uniform text cleansing.

Method 2: Handling Multiple Substrings Using Nesting

In scenarios where data requires more aggressive cleaning, you may encounter cells containing two or more distinct substrings that must be eliminated simultaneously. Since the SUBSTITUTE function is designed to perform only one replacement operation at a time, removing multiple substrings requires a technique called function nesting. Nesting involves placing one function inside another, where the output of the inner function serves as the input for the outer function.

Using the output of the first substitution as the input for the second ensures a sequential cleaning process. The innermost SUBSTITUTE function cleans the original string first, removing the first target substring. The result of this operation (the partially cleaned string) is then passed to the next SUBSTITUTE function, which removes the second target substring, and so on. This chain allows for the systematic removal of as many different strings as necessary.

Let's consider a slightly more complicated dataset where entries include both the term "Team" and the prefix "Name." Suppose we have the following data structure:

	A	B	C	D
1	**Team**	**Points**		
2	Mavs Team Name	98		
3	Pelicans Team	100		
4	Hawks Team Name	103		
5	Cavs Team	109		
6	Rockets Team	114		
7	Spurs Team	113		
8	Nets Team Name	103		
9	Knicks Team	98		
10	Kings Team Name	106		
11	Blazers Team	91		
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

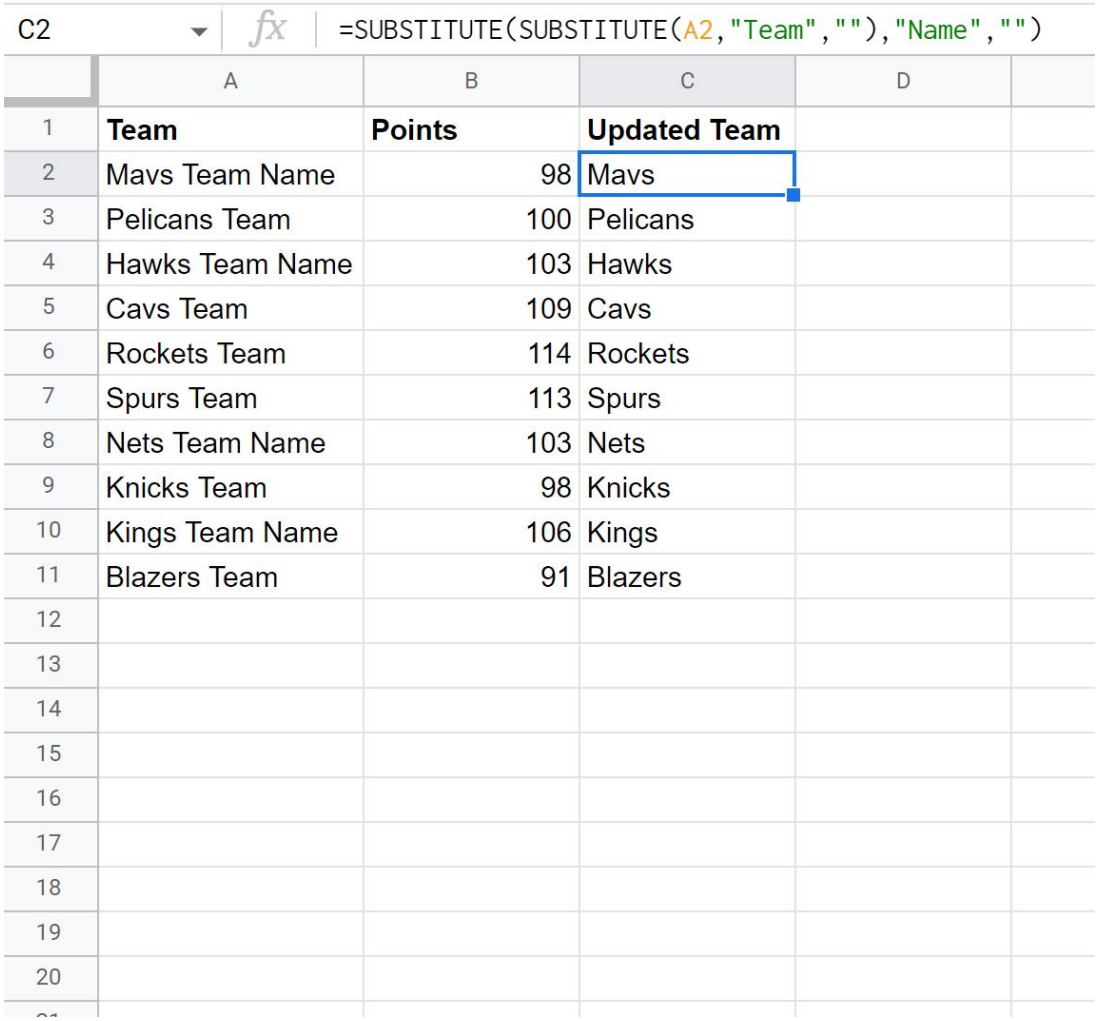
Our goal now is to remove both "Team" and "Name" from the entries in Column A. To achieve this, we will nest two SUBSTITUTE functions. The structure begins by targeting the original cell (A2). The first (innermost) SUBSTITUTE removes the first substring ("Team"). The result of this first step then becomes the `text_to_search` argument for the second (outermost) SUBSTITUTE, which removes the second substring ("Name").

We implement the following nested formula to remove both the substrings "Team" and "Name" from the content of cell A2:

=SUBSTITUTE(SUBSTITUTE(A2,"Team",""),"Name","")

After entering this formula into cell **C2**, we again use the drag-and-fill method to populate the formula down the entire column. This ensures that the two-step cleaning process is executed

sequentially for every record in the dataset.



	A	B	C	D
1	**Team**	**Points**	**Updated Team**	
2	Mavs Team Name	98	Mavs	
3	Pelicans Team	100	Pelicans	
4	Hawks Team Name	103	Hawks	
5	Cavs Team	109	Cavs	
6	Rockets Team	114	Rockets	
7	Spurs Team	113	Spurs	
8	Nets Team Name	103	Nets	
9	Knicks Team	98	Knicks	
10	Kings Team Name	106	Kings	
11	Blazers Team	91	Blazers	
12				
13				
14				
15				
16				
17				
18				
19				
20				

The resulting Column C clearly shows the successful removal of both "Team" and "Name" substrings, leaving behind only the core descriptive information. A critical insight here is that you can stack or nest as many **SUBSTITUTE** functions as needed to address a long list of unwanted strings, making this method highly scalable for complex data cleaning projects. For an extremely large number of substrings (e.g., more than five), however, alternative methods involving arrays or regular expressions might offer a cleaner, more readable solution.

Best Practices and Limitations of Using SUBSTITUTE

While the **SUBSTITUTE** function is robust and easy to deploy, users should be aware of its specific operational characteristics and limitations, especially concerning case sensitivity and pattern matching. Since SUBSTITUTE performs an exact, case-sensitive match, any deviation in capitalization will cause the function to fail to identify and replace the target string. For instance, if you search for "team" but the cell contains "Team", the substitution will not occur.

If you need case-insensitive removal, you would typically combine SUBSTITUTE with functions like **LOWER** or **UPPER** to standardize the input text before the substitution occurs. Alternatively, if the data cleaning requirements are complex, involving variable patterns (like numbers, dates, or special characters), leveraging Regular Expressions (RegEx) via the **REGEXREPLACE** function is highly recommended. RegEx allows you to define complex search patterns and perform all substitutions in a single, streamlined formula, providing superior flexibility for intricate text manipulation.

Furthermore, the nesting technique, while effective for a small number of strings (two to five), can become cumbersome and prone to error if you attempt to nest ten or more functions. This leads to extremely long, difficult-to-debug formulas. In such advanced cases, moving to **REGEXREPLACE** is often the superior practice, as it can handle an array of removals without requiring deep nesting. Always check your output columns thoroughly to ensure that the removal did not inadvertently strip out necessary characters if the substring was part of a larger, critical word.

Summary of Methods for Substring Removal

To summarize the methods discussed, the choice of approach depends entirely on the complexity and variability of the data you are processing. For simple, uniform cleaning tasks, the single SUBSTITUTE function is the go-to tool. For structured data requiring the removal of a fixed, small set of unwanted strings, function nesting provides a reliable solution.

The following ordered list outlines the key steps for executing these two primary methods successfully in Google Sheets:

Identify the Target: Precisely define the text (substring) you intend to remove. Remember to account for case sensitivity or use case-modifying functions if needed.

Determine the Replacement: For removal, the replacement argument must always be the empty string `""`.

Apply the Formula: Use a single **SUBSTITUTE** for one target string, or nest multiple SUBSTITUTE functions (starting with the original cell reference in the innermost function) for multiple target strings.

Execute and Validate: Apply the formula to the first cell, and then use the fill handle to process the entire column. Validate the results to ensure all unwanted substrings have been correctly replaced by blank space.

Mastering these fundamental string manipulation techniques ensures that your raw data is always optimally prepared for subsequent analysis, sorting, or visualization tasks within the spreadsheet environment.

Additional Resources for Data Cleaning

For users seeking to expand their knowledge of advanced data transformation and cleaning operations in Google Sheets, the following tutorials explain how to perform other common operations and explore more flexible functions:

How to use **REGEXREPLACE** for pattern-based removals instead of deep nesting.

Combining **TEXTJOIN** and **SPLIT** for complex text parsing scenarios.

Techniques for removing leading or trailing spaces using the **TRIM** function.

Understanding advanced conditional formatting based on text content.