How to Remove Characters from a String in Power BI

Authored by

mohammed looti

January 12, 2026

Yes, it is absolutely possible--and often necessary--to manipulate and remove characters from a string within Power BI. This crucial data preparation step is typically executed using the powerful string functions available in DAX (Data Analysis Expressions), most notably the SUBSTITUTE function. Unlike simply deleting text, this function allows developers and analysts to specify a precise character or set of characters to be targeted and then replaced with an empty value, effectively achieving removal. This capability is foundational for robust data cleansing and formatting, ensuring that special characters, unwanted prefixes, or extraneous spaces do not impede accurate analysis and visualization in Power BI.

The Foundation: DAX and String Manipulation

Data Analysis Expressions, or DAX, serves as the formula language for Power BI, enabling the creation of calculated columns, measures, and tables. While many text transformation tasks are optimally handled in the Power Query Editor before data loading, calculated columns defined using DAX are indispensable when transformations must occur dynamically within the data model itself or when creating columns based on pre-existing relationships and logic. String manipulation--the modification of text data--is a core requirement for almost any data project, as raw source data rarely arrives in a perfectly structured format suitable for direct reporting. Functions like **SUBSTITUTE**, **REPLACE**, and **TRIM** are the workhorses in this process, ensuring data consistency and integrity.

The choice between performing text manipulation in Power Query (M language) or in DAX often comes down to timing and purpose. If the character removal is a permanent, static change required for the entire dataset before analysis begins, Power Query is the preferred, more efficient route. However, if the removal logic must be part of a larger calculation, or if the calculation needs to be performed on a column that is itself a result of complex DAX logic, using the SUBSTITUTE function within a DAX calculated column is the appropriate methodology. This flexibility ensures that analysts can clean up data at any stage of the modeling process, maintaining full control over the data lifecycle.

Employing the SUBSTITUTE Function for Character Removal

The **SUBSTITUTE** function in DAX is specifically designed to replace existing text within a given string with new text. Crucially, by substituting the unwanted text with an empty string (represented by two quotation marks with nothing between them: ""), we effectively remove the target characters entirely. This is a highly efficient method for targeted character removal, especially when the text pattern to be removed is consistent across the column.

The standard syntax for the **SUBSTITUTE** function requires three mandatory arguments and one optional argument:

Text: This is the column or string in which you want to substitute characters.
Old Text: This is the specific text or substring you wish to find and replace. This value is case-sensitive.
New Text: This is the text that will replace the **Old Text**. To remove the characters, this argument must be specified as `""` (an empty string).
Instance Number (Optional): This argument specifies which specific occurrence of the **Old Text** should be replaced. If omitted, **all** occurrences of the **Old Text** will be replaced within the input string. For character removal scenarios, leaving this argument blank is usually desired to ensure comprehensive data cleansing across the entire cell value.

Understanding this structure is vital for successful implementation. By setting the **New Text** parameter to blank, the logic shifts from replacement to outright deletion, providing a clean and valid method for data hygiene directly within your Power BI data model.

DAX Syntax for Targeted Removal

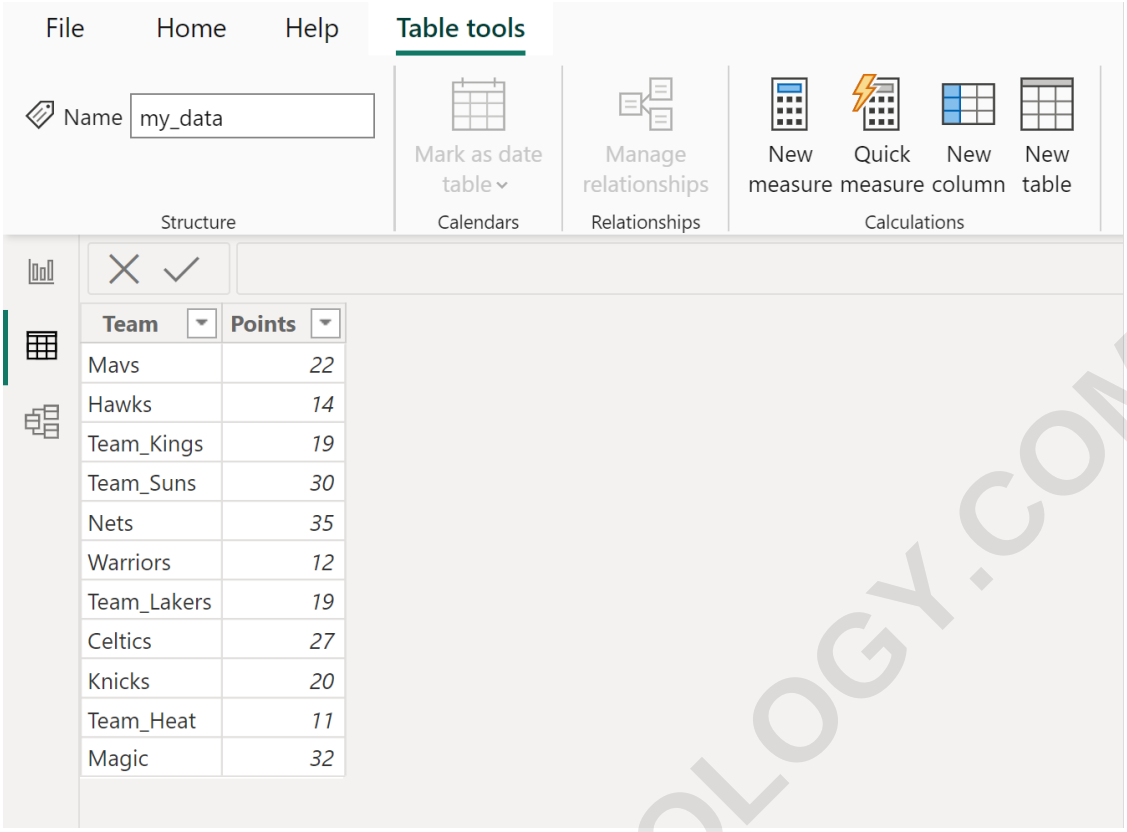
To implement this character removal technique in a Power BI report, you must create a new calculated column within your data table. The following syntax demonstrates the standard structure used to define this new column, highlighting how the **SUBSTITUTE** function is called to perform the desired action of removing the prefix "Team_" from the original data:

Team_New = SUBSTITUTE('my_data', "Team_", "")

This particular example achieves several objectives simultaneously. First, it defines a new calculated column named **Team_New** within the data model. Second, it calls the **SUBSTITUTE** function to iterate through every row in the **Team** column of the table named **my_data**. For each row, the function identifies every instance of the literal string "Team_" and replaces it with an empty string (""). This crucial substitution effectively removes the unwanted prefix, yielding a clean team name in the resulting column. This method is highly effective for standardizing categorical data that may contain unwanted identifiers or formatting prefixes.

Practical Example: Cleaning Team Data in Power BI

To fully grasp the utility of the **SUBSTITUTE** function, consider a common scenario encountered during the initial stages of data modeling. Suppose we are working with a table in Power BI, named **my_data**, which contains essential metrics related to basketball player performance, including points scored and their associated team. However, the data source has prefixed the team names with a redundant identifier, making the column less clean for visualization purposes. This structure is illustrated below, where the **Team** column clearly shows the unwanted prefix:



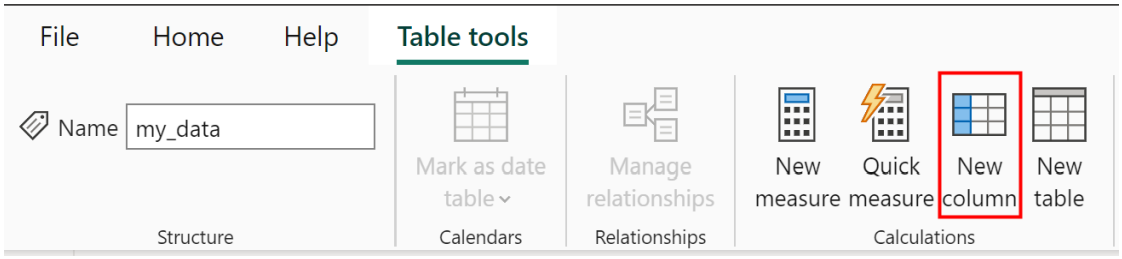
Team	Points
Mavs	22
Hawks	14
Team_Kings	19
Team_Suns	30
Nets	35
Warriors	12
Team_Lakers	19
Celtics	27
Knicks	20
Team_Heat	11
Magic	32

Upon reviewing this table, it is immediately apparent that several entries in the **Team** column contain the specific, unwanted string "Team_" embedded within the actual team name. While this prefix might have been useful in the source system, it introduces unnecessary noise into the analytical environment. For reporting and filtering purposes, we need a clean column that displays only the meaningful team name. Therefore, the goal is to create a new, derived column that systematically removes "Team_" from every applicable entry in the original **Team** column, thereby standardizing the data for aggregation and reporting.

Step-by-Step Implementation of the New Column

The process of adding a calculated column in Power BI Desktop is straightforward and utilizes the interface tools provided under the modeling ribbon. To initiate the creation of the clean team name column, we must first instruct Power BI to calculate a new field based on our DAX formula. This ensures that the original source data remains untouched while the cleaned version is readily available for reporting.

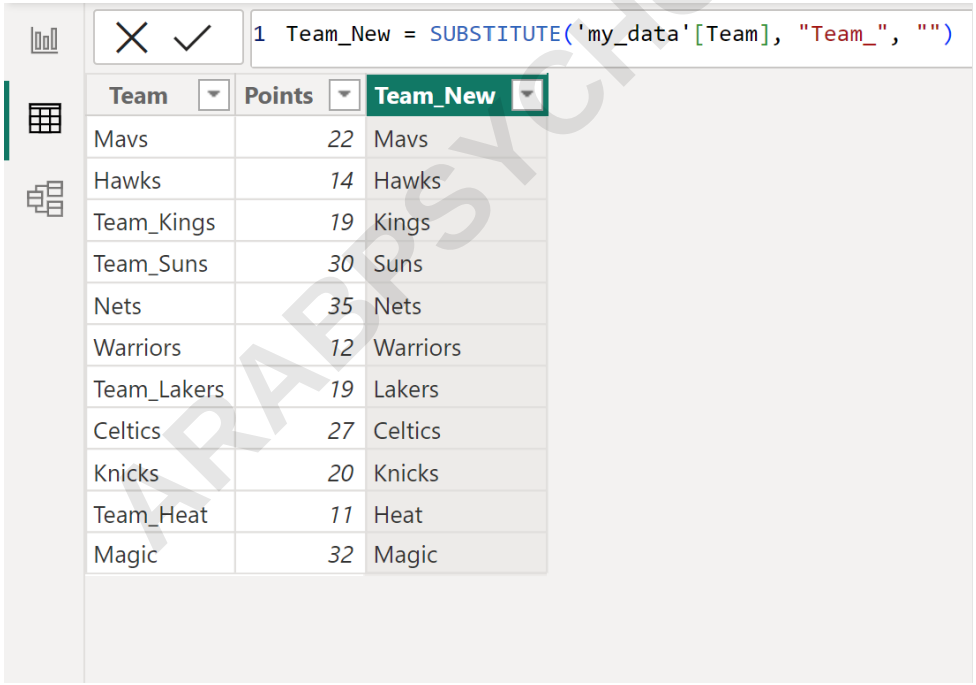
Navigate to the **Table tools** tab located along the top ribbon in Power BI Desktop.
Click on the **New column** icon. This action opens the formula bar, prompting the user to define the expression for the new column.



Once the formula bar is active, the following DAX formula is entered. This formula employs the **SUBSTITUTE** function to precisely locate and remove the targeted text: "Team_".

Team_New = SUBSTITUTE('my_data', "Team_", "")

Executing this formula instantly creates the new column named **Team_New**. This derived column now contains the cleaned team names, free of the distracting "Team_" prefix. The resulting table clearly illustrates the success of the character removal operation, showcasing standardized team names that are ready for sophisticated analysis and visualization. This is a fundamental step in ensuring the highest quality of data cleansing within your data model.



Team	Points	Team_New
Mavs	22	Mavs
Hawks	14	Hawks
Team_Kings	19	Kings
Team_Suns	30	Suns
Nets	35	Nets
Warriors	12	Warriors
Team_Lakers	19	Lakers
Celtics	27	Celtics
Knicks	20	Knicks
Team_Heat	11	Heat
Magic	32	Magic

Deep Dive into the SUBSTITUTE Formula Mechanics

The effectiveness of the character removal technique relies entirely on the precise execution of the SUBSTITUTE function. Returning to the formula used:

Team_New = SUBSTITUTE('my_data', "Team_", "")

The function is designed to handle textual substitution, making it superior to the **REPLACE** function for this specific use case. While **REPLACE** requires the user to specify the starting position and the number of characters to remove, **SUBSTITUTE** works based on the content of the string itself. This content-based approach means it does not matter where "Team_" appears in the cell; if it exists, it is targeted.

In our scenario, the function is instructed to look at the entire **'my_data'** column. It searches for `"Team_"`, and crucially, substitutes every instance it finds with `""`. The act of substitution with an empty string is what differentiates removal from modification. If we had replaced `"Team_"` with `"Club-"`, the result would be a modified string; by replacing it with nothing, the characters are seamlessly deleted, pulling the remaining parts of the string together. This robustness makes **SUBSTITUTE** the primary tool for cleaning consistent text errors or prefixes within categorical variables.

Alternative Approach: Power Query (M Language)

While DAX is essential for calculated columns, the Power Query Editor, which uses the M language, is often the more appropriate environment for initial data cleansing. Power Query transformations are applied at the source ingestion stage, meaning the resulting clean data is loaded into the data model, saving resources and improving query performance compared to relying solely on DAX calculated columns.

To remove characters using Power Query, the process is entirely graphical and requires no code input for simple replacements:

Right-click the column header (e.g., **Team**) in the Power Query Editor.
Select **Replace Values**.
In the dialog box, enter the value to find (e.g., "Team_") and leave the **Replace With** field completely blank.

Power Query generates M code behind the scenes to perform this removal, achieving the exact same result as the SUBSTITUTE function in DAX but applying the transformation earlier in the pipeline. Analysts should prioritize Power Query for static, source-level data cleanup, reserving DAX for dynamic calculations that depend on the existing data model structure.

Considerations and Best Practices for Text Transformation

When performing character removal in Power BI, there are several key considerations that impact accuracy and performance. Understanding these nuances ensures that the data cleansing process

is robust and scalable.

Case Sensitivity and Consistency

The SUBSTITUTE function in DAX is inherently case-sensitive. If your data contained "team_" in some entries and "Team_" in others, simply specifying `"Team_"` in the formula would only clean the latter, leaving the former untouched. To address inconsistencies in casing, you would need to use a nested approach that cleans both possibilities, often by converting the text to a uniform case first (using functions like **UPPER** or **LOWER**) before running the substitution.

Handling Multiple Characters

If you need to remove several different characters or substrings from the same column (e.g., removing "Team_", "Club-", and brackets ""), you must use nested **SUBSTITUTE** functions. For instance, to remove two different prefixes, the formula would look something like this:

Cleaned = SUBSTITUTE(SUBSTITUTE('Table', "Prefix1", ""), "Prefix2", "")

This approach chains the functions, where the inner **SUBSTITUTE** cleans the data, and the outer **SUBSTITUTE** operates on the result of the inner function, ensuring comprehensive removal of all targeted elements.

Performance Implications

While calculated columns using DAX are powerful, they consume memory and processing time during data refresh and querying, as the calculation is materialized in the data model. For large datasets, excessive use of complex DAX calculated columns for fundamental text cleaning can negatively impact performance. The best practice remains to offload static data transformation tasks, such as systematic character removal, to the Power Query Editor (M language) whenever possible, thereby optimizing the data model for rapid analytical queries.

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