Extract Substring in Google Sheets (With Examples)

Authored by

Mohammed looti

November 2, 2025

Mastering Substring Extraction in Google Sheets

In the realm of modern data management and analysis, effective data cleaning and preparation are non-negotiable prerequisites. This process frequently demands the surgical isolation of specific textual components--or substrings--from larger, often unstructured text strings. The technique of substring extraction is fundamentally important for tasks such such as standardizing identifiers, separating metadata, or isolating critical keywords within massive datasets.

Google Sheets is equipped with a powerful and flexible array of text manipulation functions designed to handle these complex requirements with efficiency. Analysts must move beyond simple sorting and filtering to embrace dynamic text functions that allow for the intelligent reshaping of variable data formats.

This tutorial will dissect five primary methods for extracting text. We categorize these into two strategic approaches: **position-based extraction**, which relies on fixed character counts from the beginning or end of the string, and **dynamic extraction**, which uses locator text or specific delimiters to identify the target text regardless of its position or length. Success in these methods hinges on mastering the core functions: **LEFT**, **MID**, and **RIGHT**, often combined with the robust locator function, **SEARCH**, to tackle highly variable data structures.

The Foundation: Position-Based Extraction Functions

Position-based extraction offers the most direct solution when the source data adheres to a strict, consistent format. If the segment you need always begins at the fifth character or always spans exactly ten characters, these functions provide speed and simplicity. They require the user to define the exact character count, starting either from the string's origin or its conclusion.

A critical foundational concept in Google Sheets text manipulation is character indexing. Every character in a cell, including spaces, punctuation marks, and special characters, is counted as a single position. Therefore, accurate character counting is absolutely essential for achieving precise results when using position-based functions.

Below, we detail the three fundamental functions that form the backbone of fixed-position extraction across any given text string.

Method 1: Extracting from the Start (The LEFT Function)

The LEFT function is the designated tool for retrieving a specified number of characters counting from the first position of the string. This is particularly valuable when standard prefixes--such as product codes, regional identifiers, or standardized date formats--consistently reside at the start of the cell's content.

The function is streamlined, requiring only two arguments: the source text **string** (often a cell reference like A1) and the **number_of_characters** you intend to extract. The syntax is simply `=LEFT(string, number_of_characters)`. It is worth noting that if the requested number of characters exceeds the total length of the source string, the function will gracefully return the entire string without error.

#return first 4 characters of string in cell A1
=LEFT(A1, 4)

Utilizing **LEFT** is the most straightforward and computationally efficient way to segment data when the length of the prefix is guaranteed to remain constant across all relevant entries in your spreadsheet.

Method 2: Isolating Embedded Data (The MID Function)

When the target substring is located centrally, sandwiched between preceding and trailing text, the MID function becomes indispensable. This function is often considered the most flexible of the positional tools because it provides granular control over both the extraction starting point and the total length of the segment to be pulled.

The **MID** function demands three critical arguments: the source **string**, the **starting_position** (where extraction begins, indexed from 1), and the **number_of_characters** to extract from that starting point. For instance, to start at the second character and extract four subsequent characters, you would input `=MID(A1, 2, 4)`.

#return 4 characters of string in cell A1 starting at position 2
=MID(A1, 2, 4)

A useful behavior of **MID** is its handling of boundaries: if the designated starting position plus the extraction length extends beyond the string's end, **MID** will simply return all available characters until the text concludes. This makes it highly effective for extracting embedded identifiers in database exports where the segment is fixed in position but might vary slightly in length if it's near the end of the field.

Method 3: Capturing the End Segment (The RIGHT Function)

To efficiently isolate text that terminates a string--such as file extensions, version numbers, or trailing suffixes--the RIGHT function is the appropriate choice. This function operates analogously to **LEFT**, but it performs the character count backward, starting from the final character of the text.

The syntax perfectly mirrors the **LEFT** function: `=RIGHT(string, number_of_characters)`. This

function is extremely efficient because it eliminates the need to calculate the total length of the preceding text, allowing for rapid capture of trailing data segments.

#return last 4 characters of string in cell A1
=RIGHT(A1, 4)

When deploying the **RIGHT** function, the analyst must confirm that the desired data segment is consistently positioned at the very end of the string. If the trailing segment is inconsistent or variable in nature, relying purely on fixed position extraction may introduce errors or yield incomplete results.

Dynamic Extraction Using Locator Functions (The Power of SEARCH)

In the vast majority of real-world data scenarios, text strings do not conform to fixed lengths or positions. Consider the common task of extracting a username from an email address; the length of the username varies, requiring a mechanism that can dynamically locate the "@" symbol, which acts as the delimiter, and then calculate the variable length of the required substring.

To achieve this flexibility, we must elevate the positional functions (LEFT, MID, RIGHT) by nesting them with a locator function. The most prevalent and powerful locator in Google Sheets is the SEARCH function, which scans a string to determine the precise starting position of a specified character or sequence of characters.

By leveraging **SEARCH**, we transition our formulas from static, fixed-position operations to dynamic calculations. This capability is crucial for achieving accurate and scalable data segmentation across entire columns of text that may be highly inconsistent in their formatting.

Method 4: Extracting Text Before a Specific Delimiter

A fundamental application of dynamic extraction is isolating all text that precedes a known marker or delimiter. This is elegantly accomplished by nesting the **SEARCH** function within the **LEFT** function. Here, **SEARCH** is used not to return the substring itself, but to calculate the exact stopping point for **LEFT**.

To ensure the delimiter itself is properly excluded from the resulting output, it is necessary to subtract one unit from the position number returned by **SEARCH**. The resulting structure is highly effective: `=LEFT(string, SEARCH("marker", string) - 1)`. This instructs the formula to pull characters up to, but not including, the delimiter.

#return all text before the string "there" in cell A1
=LEFT(A1, SEARCH("there", A1)-1)

This dynamic approach is indispensable for rapidly splitting compound data fields, such as separating product names from associated descriptive text where a space, hyphen, or underscore serves as the consistent separator.

Method 5: Extracting Text After a Specific Delimiter

Extracting the text that follows a specified delimiter demands a more sophisticated mathematical approach, as it requires calculating the variable remaining length of the string after the marker. This often necessitates combining the **RIGHT** function with **LEN** (Length) and SEARCH.

The core logic involves three steps: first, find the total length of the string using **LEN()**. Second, find the starting position of the delimiter using **SEARCH()**. Third, subtract the delimiter's position (and its length) from the total string length. The resulting difference gives us the precise length of the trailing text, which is then fed into the RIGHT function to complete the extraction.

```
#return all text after the string "there" in cell A1
=RIGHT(A1, LEN(A1) - SEARCH("there", A1) - 4)
```

While the exact formula structure requires careful adjustment based on the length of the delimiter itself (here, 'there' is 5 characters, so the subtraction must account for that length), the underlying principle remains unwavering: use **SEARCH** to establish the anchor point, and then utilize length calculations (via **LEN** and **RIGHT** or **MID**) to accurately capture the remaining, variable portion of the string.

Visualizing Extraction Methods in Practice

To solidify your understanding of these techniques, the following examples provide clear, practical demonstrations of how each of the five extraction methods functions within the Google Sheets environment. These visualizations are based on applying the specific formulas discussed above to a consistent sample text string located in cell A2.

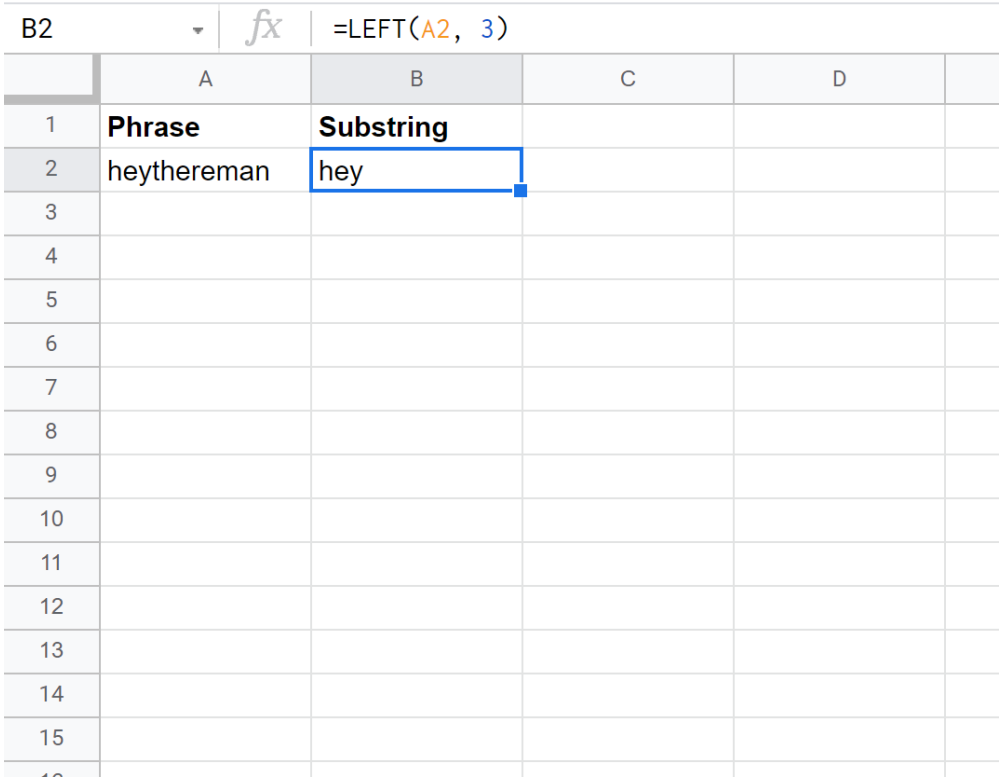
Careful review of these screenshots helps reinforce the critical nuances of character counting and position indexing--skills that are necessary for truly effective data manipulation and data cleaning.

Method 1 Example: Extracting the Prefix using LEFT

This illustration clearly depicts the result of applying the LEFT function to retrieve the first three characters from the source text. This serves as the textbook example of position-based extraction where the desired segment length is fixed and calculated strictly from the beginning of the string.

Note how the function cleanly isolates the initial three characters, subsequently discarding all

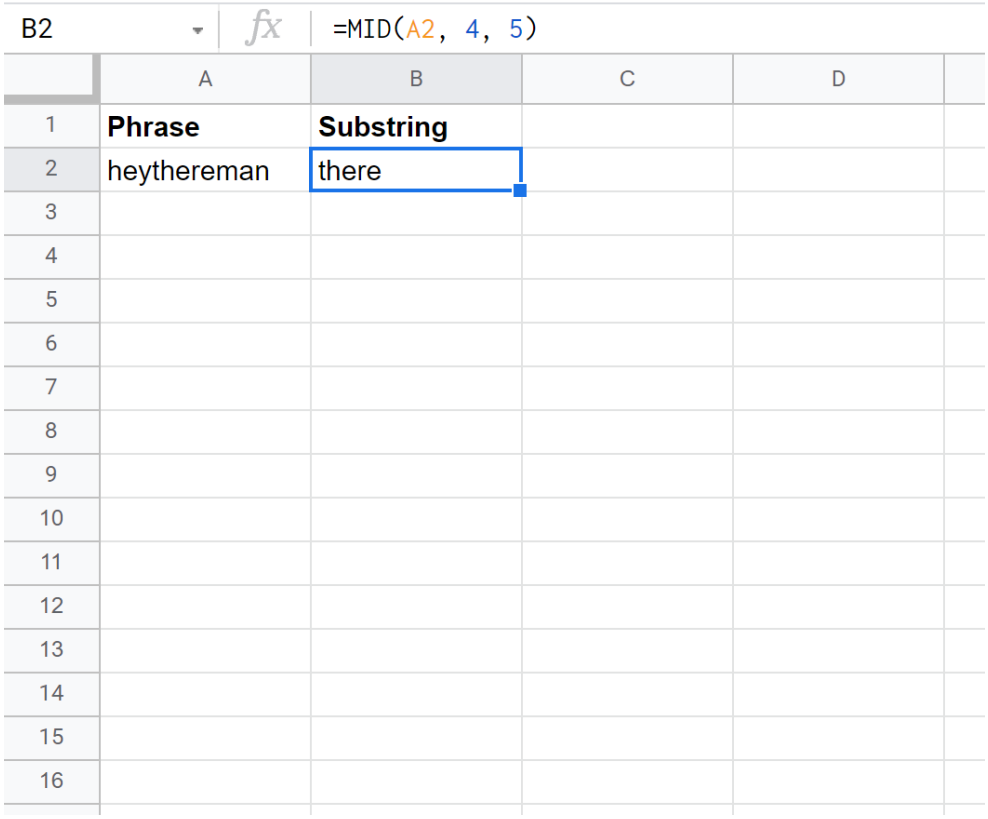
trailing content within the cell.



Method 2 Example: Isolating Embedded Data using MID

The screenshot below powerfully demonstrates the utility and precision of the MID function. In this scenario, the applied formula extracts exactly five characters, commencing precisely at the fourth character position of the string found in cell A2.

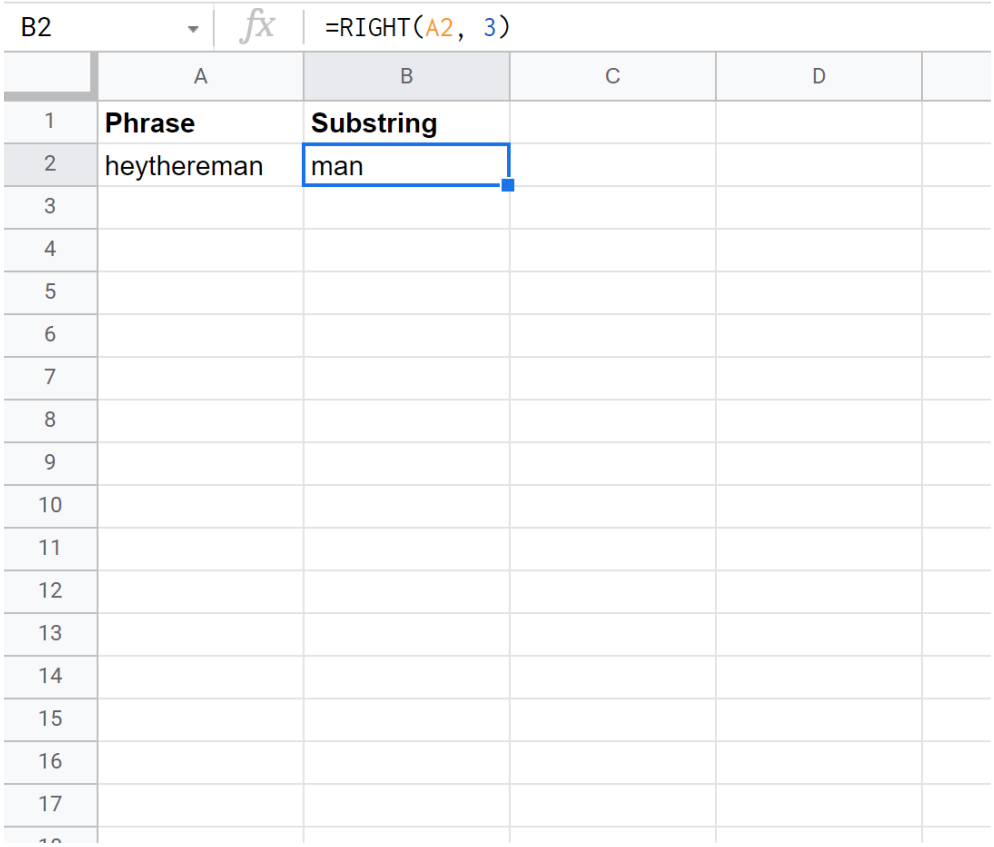
This technique is vital when the key data segment is embedded, requiring the definition of both a precise starting anchor and a specific, fixed length for successful extraction.



Method 3 Example: Retrieving Suffixes using RIGHT

In this final positional example, the RIGHT function is deployed to retrieve the concluding three characters from the text contained in cell A2. This confirms the function's mechanism of counting backward, enabling the isolation of the terminal segment regardless of the total length of the preceding text.

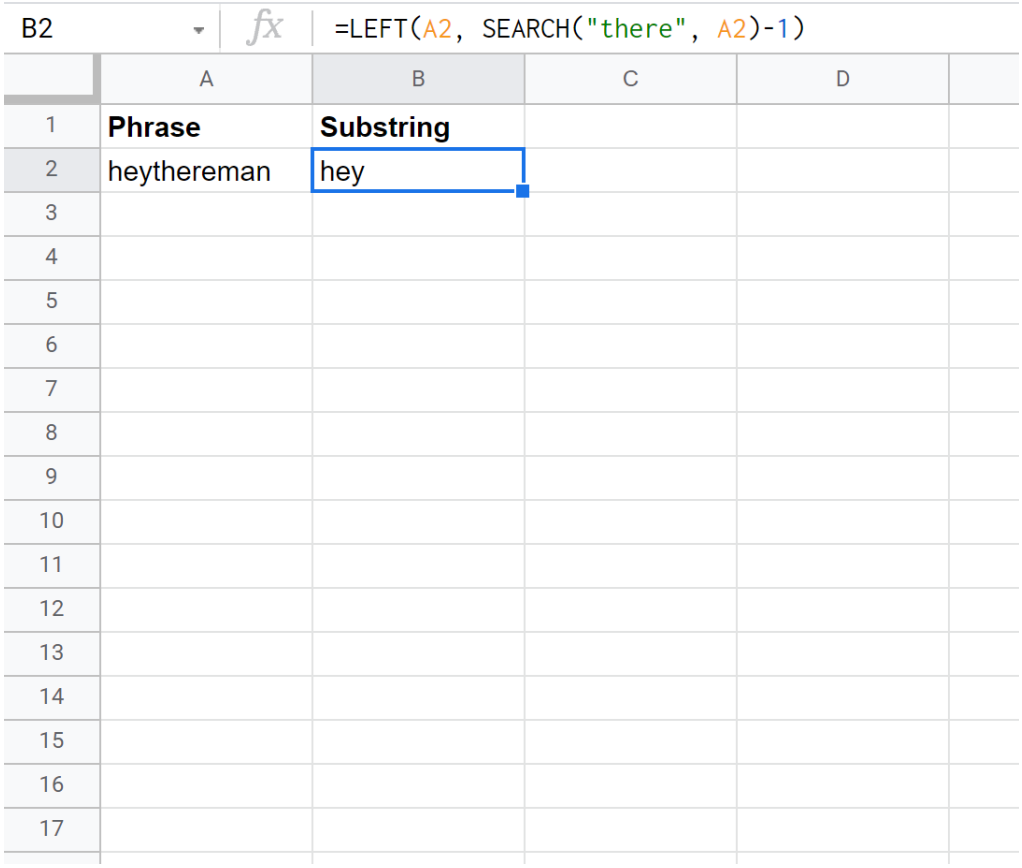
This method drastically simplifies data cleanup tasks associated with consistent trailing information, such as standardized currency codes or unit measurements.



Method 4 Example: Dynamic Extraction Before a Delimiter

This visualization showcases the combined power of **LEFT()** and SEARCH() used in tandem to dynamically extract all text located before the specific word "there" in cell A2. Observe carefully how the resulting string is accurately truncated immediately preceding the first character of the identified search term.

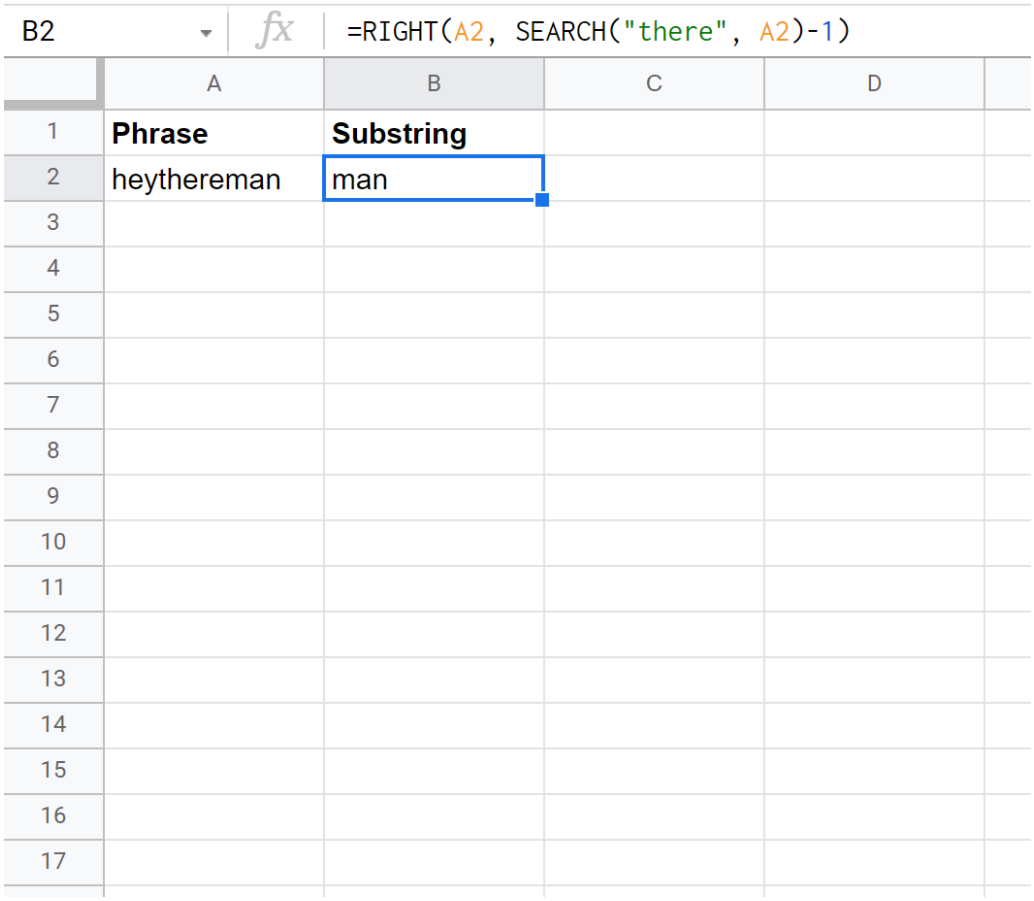
This is recognized as a highly efficient and adaptable method for standardizing data by splitting varied entries based on a consistent, but non-fixed, marker.



Method 5 Example: Dynamic Extraction After a Delimiter

The concluding example demonstrates the extraction of text that immediately follows the marker "there" in cell A2, achieved by complex calculation utilizing the **RIGHT()** and SEARCH() combination. The formula successfully and dynamically calculates the necessary length of the remaining string starting precisely after the specified anchor marker.

This advanced method is essential for isolating descriptive text or crucial values that appear after a standardized field name or header within a single, composite cell.



Conclusion and Next Steps in Text Processing

The capability to accurately and efficiently extract a substring in Google Sheets stands as a critical cornerstone of advanced data management and spreadsheet proficiency. By thoroughly mastering the five methods presented--which leverage **LEFT**, **MID**, and **RIGHT** for predictable, fixed positions, and strategically combine them with **SEARCH** for dynamic, delimiter-based segmentation--you gain precise, automated control over text manipulation.

These sophisticated techniques ensure that your raw data is consistently and correctly formatted, making it perfectly suitable for subsequent rigorous analysis, integration into database systems, or streamlined reporting. Analysts should always evaluate the consistency of their source data before selecting an extraction method, prioritizing positional methods for fixed data and dynamic methods for variable data.

To further refine your text processing skills within spreadsheet environments, consider exploring tutorials and documentation covering these related essential operations:

Explore functions used to combine multiple text strings (known as **concatenation**) using the CONCATENATE function or the ampersand operator (&).

Learn about the enhanced capabilities of regular expressions (specifically the **REGEXEXTRACT**, **REGEXMATCH**, and **REGEXREPLACE** functions) for tackling highly complex and unpredictable pattern matching requirements.

Review essential methods for maintaining data integrity, such as utilizing **TRIM** to eliminate excess whitespace or **LOWER/UPPER** to standardize text case.