Extracting Text Between Quotes in Excel: A Step-by-Step Guide Using TEXTBEFORE and TEXTAFTER

Authored by

Mohammed looti

November 9, 2025

Introduction: Mastering Text Extraction in Excel

The ability to effectively clean and manipulate raw data is a cornerstone of professional spreadsheet management. A recurring and often tedious challenge faced by users of Excel is the precise isolation of specific text strings that are delimited by punctuation, most notably quotation marks. In the past, achieving this level of surgical extraction demanded the creation of convoluted, error-prone formulas involving complex nesting of older functions like **MID**, **FIND**, and **SEARCH**, requiring meticulous calculation of character positions and string lengths.

Fortunately, modern updates to the platform have introduced highly efficient, intuitive text functions that dramatically simplify this process. Key among these are **TEXTBEFORE** and **TEXTAFTER**. These functions eliminate the need for manual character counting and provide a robust method to extract all text strictly positioned between a pair of quotes--be they double quotes ("") or single quotes (')--using a straightforward, nested formula structure. This advancement transforms a previously complex task into a routine operation.

The true utility of the TEXTBEFORE and TEXTAFTER functions lies in their dynamic capability. They effortlessly adapt to data where the lengths of strings and the positions of delimiters vary widely. Throughout this guide, we will dissect the two primary methodologies for coupling these functions to achieve perfect, reusable text extraction solutions suitable for diverse datasets.

The Extraction Mechanism: How TEXTBEFORE and TEXTAFTER Work Together

To successfully extract a substring that is flanked by two identical boundary characters (the opening and closing quotes), we must deploy a strategic nested approach. This technique isolates the target text by using the two functions sequentially to define the exact start and end boundaries, effectively filtering out all surrounding noise, or prefix and suffix data. This procedure can be best conceptualized as a two-stage digital trimming process.

The process initiates with the **TEXTAFTER** function, which operates as the core, innermost component of the formula. This function targets the original source cell (e.g., A2) and returns all content that immediately follows the first encounter of the specified delimiter (the opening quote). The immediate result of this step is an interim string that begins with the desired text, followed by the closing quote and any trailing characters that remain.

This interim result is then instantly passed to the outer **TEXTBEFORE** function. Acting as the final filter, **TEXTBEFORE** treats the interim string as its source text, locates the closing quote delimiter, and efficiently returns only the characters that precede it. This synergistic application--using **TEXTAFTER** to define the starting point and **TEXTBEFORE** to define the stopping point--guarantees that only the data precisely positioned between the delimiters is isolated and returned.

It is important to note that the required formula syntax changes slightly depending on whether you are targeting double quotes or single quotes. This variation is necessary due to how Excel internally processes and interprets character strings used as delimiters within its own formula syntax.

Syntax Essentials: Handling Delimiters

Specifying the delimiter correctly is paramount for executing a successful extraction. Since the quotation mark character serves dual purposes--it acts as the boundary in your source data and also as a syntax identifier for defining text strings within the Excel formula itself--special handling techniques, known as "escaping," are required to instruct the program to look for the literal character.

Method 1: Extract Text Between Double Quotes ("")

When focusing on text enclosed by **double quotes**, the formula necessitates a specific syntax to escape the character. To communicate to Excel that the delimiter you are searching for is a literal double quote, you must enclose two double quotes within the standard formula's double quotes. This complex structure results in the delimiter argument being represented by `""""`. This optimized formula structure provides a reliable solution for extracting elements from standard text strings where double-quoted information is common.

=TEXTBEFORE(TEXTAFTER(A2, """"), """")

Method 2: Extract Text Between Single Quotes (')

Extracting text bounded by **single quotes** (often referred to as apostrophes) is significantly less syntactically demanding. Because the single quote does not function as an inherent reserved formula character in the same way the double quote does, the delimiter argument can be specified in a straightforward manner: a single quote enclosed within the standard formula double quotes, represented as `"'"`. This simplified representation enhances the accessibility and ease of use when working with data that utilizes single-quoted boundaries.

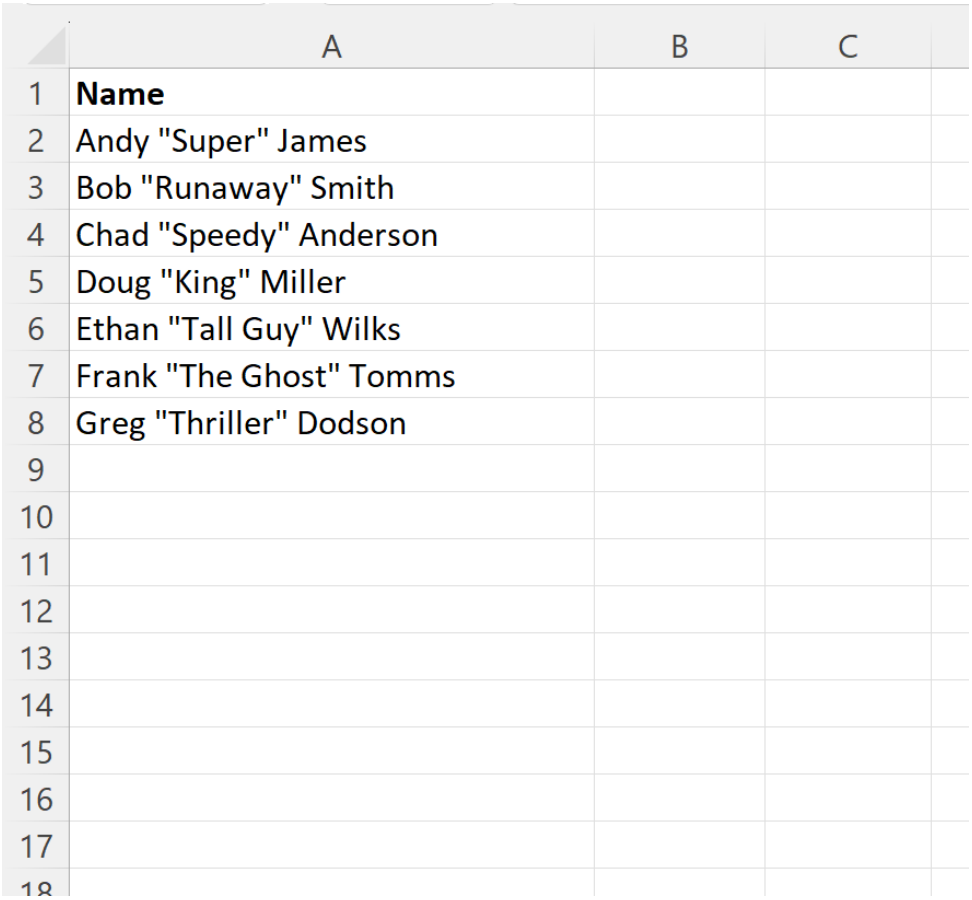
=TEXTBEFORE(TEXTAFTER(A2, "'"), "'")

In both essential syntax variants, the critical nested structure successfully targets and isolates the required text from the source cell, **A2**, thereby confirming the consistent and powerful application of **TEXTAFTER** followed by **TEXTBEFORE** to define precise extraction boundaries.

Practical Example 1: Extracting Content Bounded by Double Quotes

Imagine a typical scenario in data cleaning where a raw data source provides a list of individuals, and their frequently used nicknames or aliases are embedded within the primary text, clearly distinguished by double quotation marks. Our objective is to efficiently generate a clean, isolated list of only the nicknames in a separate column for subsequent analysis or reporting.

Assume Column A contains the structured text entries as displayed below, where the nicknames are encased in **double quotes**:



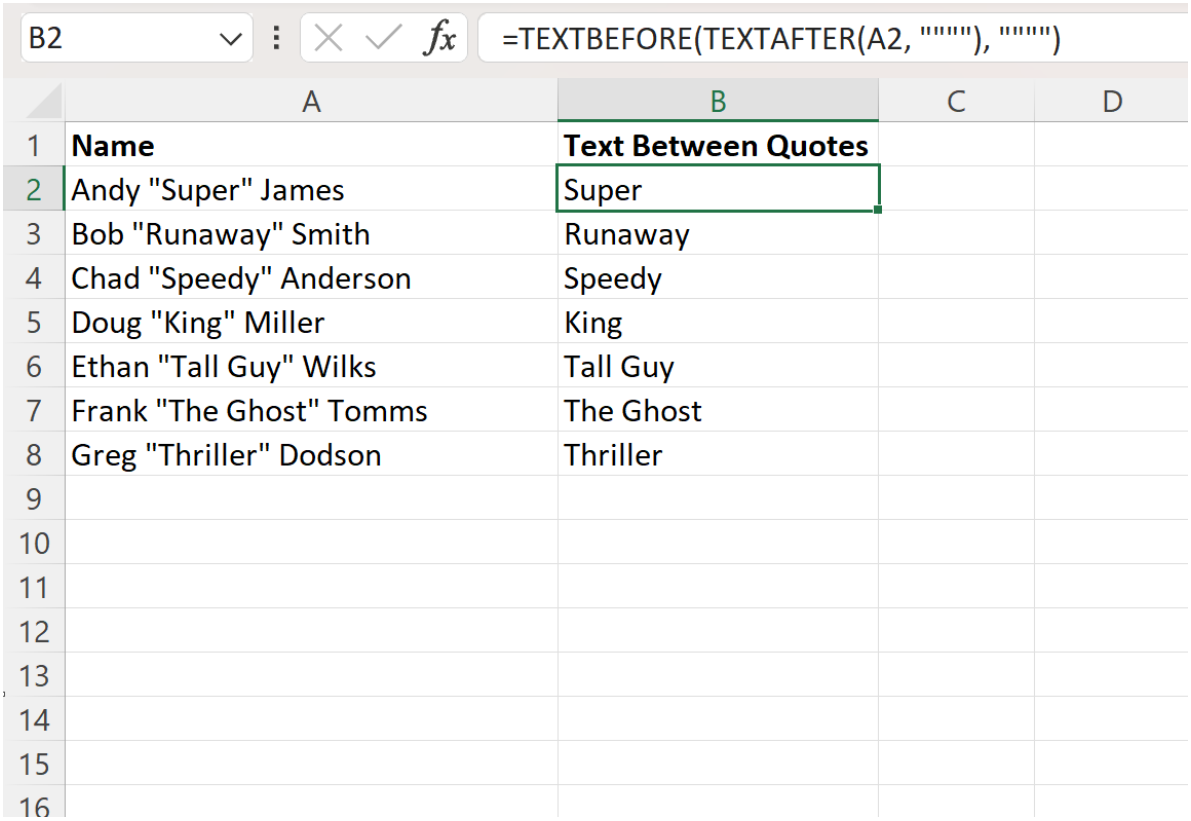
	A	B	C
1	**Name**		
2	Andy "Super" James		
3	Bob "Runaway" Smith		
4	Chad "Speedy" Anderson		
5	Doug "King" Miller		
6	Ethan "Tall Guy" Wilks		
7	Frank "The Ghost" Tomms		
8	Greg "Thriller" Dodson		
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

To perform the necessary extraction for the first entry (located in cell A2), we must use the formula variant specifically designed for **double quotes**. We input the required formula into cell **B2**. The inner TEXTAFTER function first executes its task, stripping away all characters preceding and including the opening delimiter `"""`. Subsequently, the outer TEXTBEFORE function takes the resulting substring and uses the same delimiter to halt the string extraction just before the closing `"""`.

The specific formula entered into cell **B2** is:

=TEXTBEFORE(TEXTAFTER(A2, """"), """")

Once this formula successfully retrieves the nickname for the initial row, efficiency is maximized by leveraging Excel's fill handle feature. By simply clicking and dragging the formula down to the corresponding cells in Column B, the relative cell reference (A2) automatically adjusts for each row (A3, A4, etc.). This ensures the identical, accurate extraction logic is applied across the entire dataset instantly, eliminating any need for manual intervention or verification of character counts.



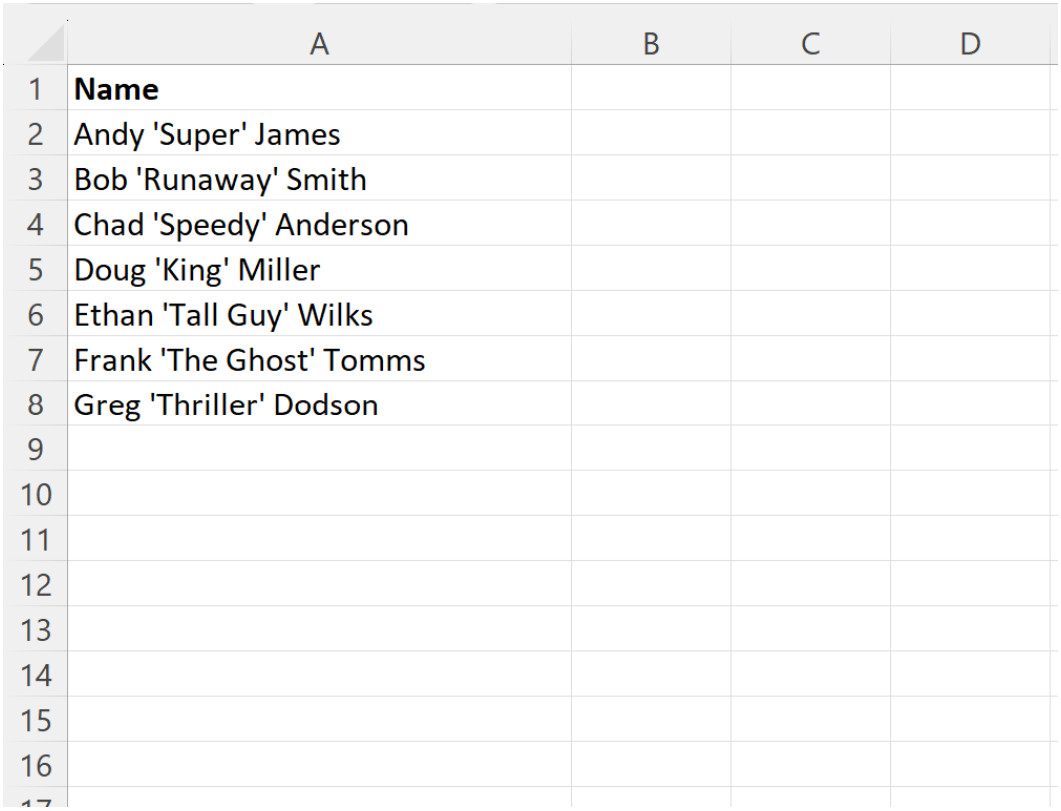
	A	B	C	D
1	**Name**	**Text Between Quotes**		
2	Andy "Super" James	Super		
3	Bob "Runaway" Smith	Runaway		
4	Chad "Speedy" Anderson	Speedy		
5	Doug "King" Miller	King		
6	Ethan "Tall Guy" Wilks	Tall Guy		
7	Frank "The Ghost" Tomms	The Ghost		
8	Greg "Thriller" Dodson	Thriller		
9				
10				
11				
12				
13				
14				
15				
16				

The resulting output in Column B clearly confirms that the nicknames have been isolated with high precision, validating the effectiveness and robustness of the nested formula when dealing with data delimited by **double quotes**.

Practical Example 2: Working with Single Quotes

In many situations, particularly when dealing with data exported from specific database environments or scripting languages, **single quotes** (') are utilized as delimiters instead of double quotes. While the visual delimiter changes, the core methodology for extraction remains fundamentally identical; however, we must switch to the simpler single-quote syntax outlined earlier.

Consider an analogous list of structured text, but this time, the athlete nicknames are specifically bounded by single quotes, as shown in the source data below:



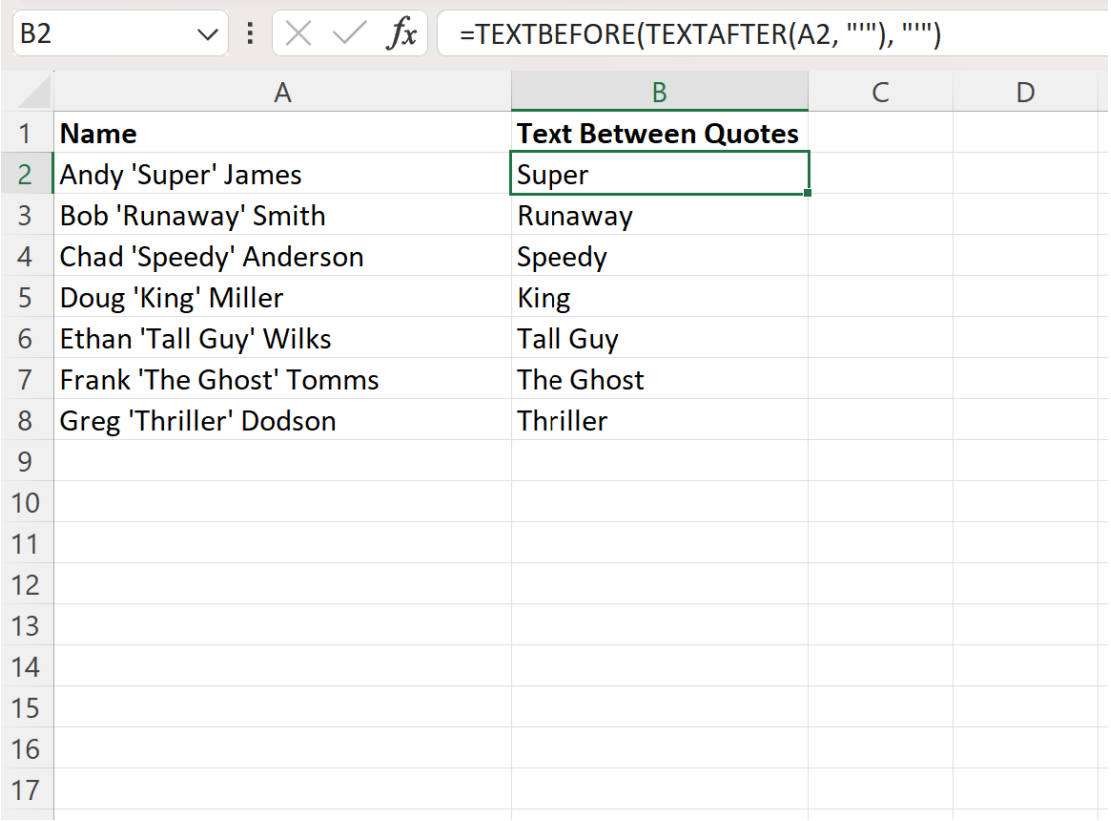
	A	B	C	D
1	**Name**			
2	Andy 'Super' James			
3	Bob 'Runaway' Smith			
4	Chad 'Speedy' Anderson			
5	Doug 'King' Miller			
6	Ethan 'Tall Guy' Wilks			
7	Frank 'The Ghost' Tomms			
8	Greg 'Thriller' Dodson			
9				
10				
11				
12				
13				
14				
15				
16				

To isolate the text segments bounded by **single quotes**, we again rely heavily on the powerful nested structure, but we specify the delimiter as `"'"`. This specification explicitly instructs both text functions to search for and use the single apostrophe character as the boundary marker. As before, the inner **TEXTAFTER** function discards all preceding content up to the opening single quote, and the outer **TEXTBEFORE** function then extracts the desired substring that precedes the closing single quote.

We input the following formula into cell **B2**:

=TEXTBEFORE(TEXTAFTER(A2, "'"), "'")

Once the calculation is correctly established in the first cell, applying this logic to the remainder of the column is quick and efficient. By dragging the formula down, we guarantee that the extraction logic is consistently applied, effectively handling variations in the length of the surrounding text or the extracted content itself. This level of automation underscores the key benefit of integrating these modern text manipulation tools into your regular data cleaning workflow.



The final result confirms that Column B successfully contains only the text segments that were precisely enclosed within the **single quotes** from the corresponding cells in Column A, thereby validating the adaptability of the **TEXTBEFORE** and **TEXTAFTER** functions across different delimiter types.

Troubleshooting and Robust Data Handling

While these nested formulas offer highly elegant and efficient solutions for routine text extraction, users must remain vigilant regarding source data integrity. Inconsistencies, such as missing delimiters or the presence of multiple sets of quotes within a single cell, require specialized handling to prevent formula errors.

A fundamental requirement for the formula's success is the existence of both an opening and a closing quote. If a source cell contains the opening quote but lacks the closing quote, the **TEXTBEFORE** function will fail to locate its required delimiter and will typically return a `#VALUE!` error. This error clearly signals that the expected text boundary was not found. For robust data analysis and professional reporting, it is a best practice to wrap these extraction formulas within an **IFERROR** function. This allows you to manage cells missing delimiters gracefully, perhaps by returning a blank cell, the original text, or a specific error notification like "Boundary Missing," ensuring a clean output column.

A final, important consideration arises when the source cell contains **multiple sets of quotes** (e.g., both a name and an address are quoted). By default, the formulas demonstrated here are designed to extract the text between the **first instance** of the opening quote and the **first subsequent instance** of the closing quote. If your goal requires extracting text from the second, third, or any subsequent quoted segment, you must utilize the optional `instance_num` argument within the TEXTAFTER function. By setting the `instance_num` to 2 (e.g., `TEXTAFTER(A2, "'", 2)`), you instruct the function to begin extraction only after the second opening quote, providing the necessary customization for handling complex, multi-delimited data formats.

Conclusion: Streamlining Your Data Workflow

The introduction of modern text functions like **TEXTBEFORE** and **TEXTAFTER** represents a significant leap forward in Excel functionality. They simplify data manipulation tasks that previously demanded resource-intensive, often unreliable combinations of older string formulas. Mastering tools such as these is absolutely crucial for any user seeking to streamline data preprocessing workflows, enhance data integrity, and maximize overall spreadsheet efficiency.

The following resources provide further tutorials that explain how to perform other common text manipulation tasks in Excel, helping you expand your skillset beyond delimiter-based extraction.