How to Easily Extract Text Within Quotes in Excel

Authored by

stats writer

February 23, 2026

Excel: Extract Text Between Quotes

In the modern landscape of **data analysis**, the ability to efficiently manipulate and clean strings is a fundamental skill. When working within **Excel**, users frequently encounter datasets where essential information is encapsulated within quotation marks, such as nicknames, specific identifiers, or localized attributes. Extracting this data manually is not only prone to human error but is also incredibly time-consuming when dealing with thousands of rows. Fortunately, the introduction of dynamic array functions has revolutionized how we approach **string manipulation**, allowing for elegant and robust solutions that were previously complex to implement.

The primary tools for this task are the **TEXTBEFORE** and **TEXTAFTER** functions. These functions represent a significant upgrade over legacy methods that relied on nesting the **FIND**, **SEARCH**, and **MID** functions. By leveraging these newer capabilities, users can create formulas that are much easier to read, maintain, and troubleshoot. This article provides a comprehensive overview of how to utilize these functions to isolate text within both double and single quotes, ensuring your **data cleaning** workflow remains professional and efficient.

Whether you are a seasoned data scientist or a business professional looking to streamline your **spreadsheet** tasks, understanding the mechanics of nested text functions is invaluable. The following methods demonstrate how to strip away unwanted characters and focus strictly on the core content. By mastering these techniques, you ensure that your **Microsoft 365** environment operates as a powerful engine for data transformation, moving beyond basic entry into advanced information architecture.

Understanding the Mechanics of TEXTAFTER and TEXTBEFORE

To successfully extract text between delimiters, one must first understand the logical flow of the functions involved. The **TEXTAFTER** function is designed to return all text that occurs after a specified character or string. In our context, this function is used to identify the first quotation mark and discard everything to its left, including the quote itself. This leaves us with a string that begins exactly with the content we wish to extract, followed by the remaining text and the closing quote.

Once the initial portion of the string is isolated, the **TEXTBEFORE** function is applied to the result. This function works in the opposite direction, returning all text that appears before a designated delimiter. By setting the delimiter as the closing quotation mark, **Excel** effectively "clips" the end of the string. The result of this nested operation is the clean, isolated text that was originally trapped between the two marks. This two-step logical process is the standard for modern **algorithm** design within cell-based formulas.

The beauty of this approach lies in its versatility. While this guide focuses on quotes, the same

logic can be applied to parentheses, brackets, or any custom delimiters found in **CSV** files or **JSON** exports. Understanding the **syntax** of these functions allows you to build complex data extraction pipelines that are resilient to varying string lengths and content types, provided the delimiters remain consistent throughout the dataset.

Method 1: Extracting Text Between Double Quotes

Extracting text from double quotes requires a specific **syntax** because the double quote character itself is used to define strings within **Excel** formulas. To represent a literal double quote inside a formula, you must use an escape sequence. This is why the formula looks slightly more complex than one might expect. By using four consecutive double quotes, you are telling the **Excel** engine that you want to search for the specific character **"** rather than ending the string argument of the function.

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

In this specific configuration, the inner **TEXTAFTER** function targets cell **A2**. It scans the text until it finds the first instance of a double quote and returns everything following it. For example, if the cell contains *Athlete "The Rocket" Smith*, the inner function returns *The Rocket" Smith*. This is the first critical step in the **data processing** sequence, setting the stage for the final extraction.

Following this, the outer **TEXTBEFORE** function takes that intermediate result and looks for the next double quote. It captures everything before that quote, which in our example would be *The Rocket*. This nested structure ensures that the final output is stripped of all structural characters, leaving only the desired nickname or value. This method is highly reliable for cleaning up **database** exports where attributes are often wrapped in quotes to prevent delimiter clashing.

Method 2: Extracting Text Between Single Quotes

The process for single quotes is logically identical but syntactically simpler. Since **Excel** uses double quotes to define string literals, a single quote (apostrophe) does not need to be escaped in the same way. You simply wrap the single quote in double quotes to identify it as the delimiter. This makes the formula significantly easier to read at a glance, though it performs the exact same computational steps as the double-quote variation.

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

When this formula is executed, **Excel** looks for the first single quote in the target cell. It discards the prefix and then searches the remaining string for the subsequent single quote to discard the suffix. This is particularly useful when dealing with **SQL** queries or certain **programming**

languages where single quotes are the standard for string literal definitions. Utilizing this formula maintains **data integrity** by ensuring no extra spaces or characters are accidentally included in the output.

By using this streamlined approach, you avoid the pitfalls of the older **Regular Expression**-style logic that was often required in previous versions of the software. The **TEXTBEFORE** and **TEXTAFTER** functions are part of the **dynamic array** update, meaning they are optimized for performance and can handle large ranges of data without the significant recalculation overhead associated with more complex, volatile formulas.

Example 1: Practical Application for Double Quotes

To illustrate the power of this method, let us consider a practical scenario involving a list of professional athletes. In many sports databases, an athlete's nickname is included within the same cell as their full name, typically enclosed in double quotes. This presents a challenge for sorting or filtering by nickname. To resolve this, we can apply our formula to isolate the nicknames into a dedicated column, facilitating better **information retrieval**.

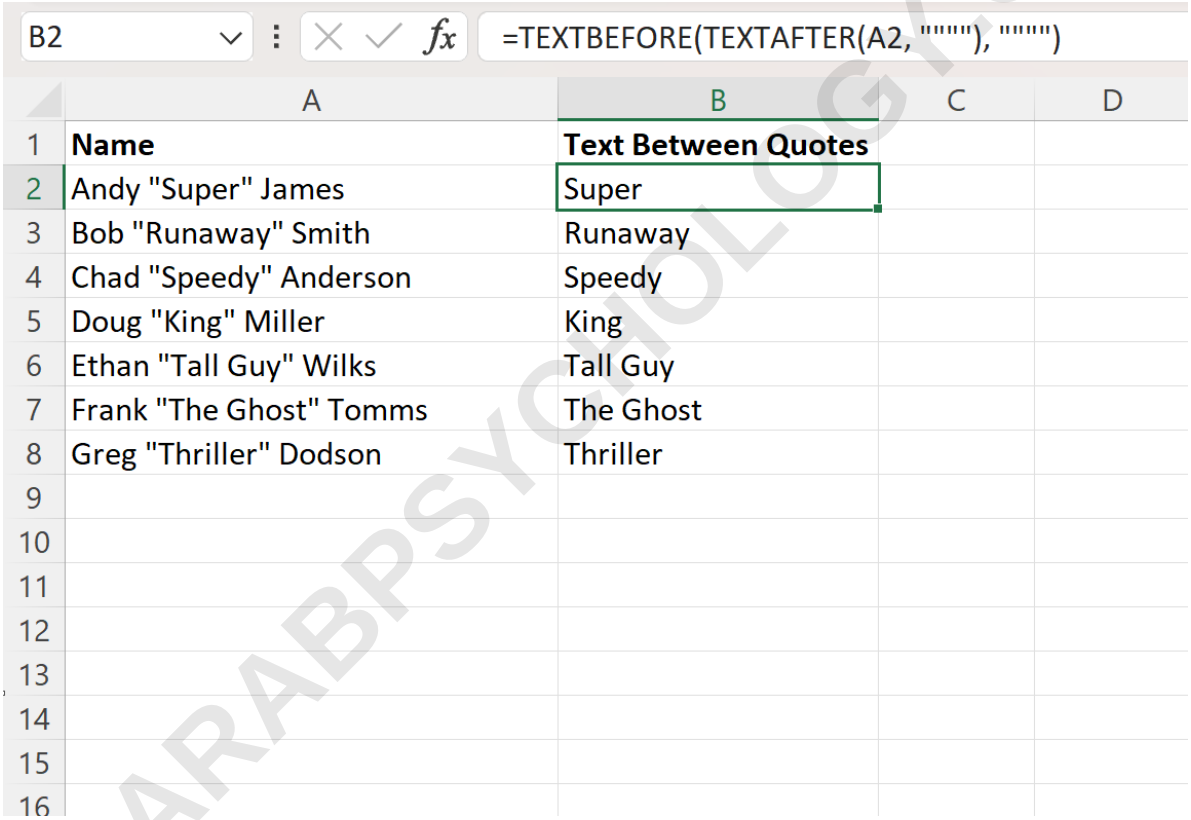
	A	B	C
1	**Name**		
2	Andy "Super" James		
3	Bob "Runaway" Smith		
4	Chad "Speedy" Anderson		
5	Doug "King" Miller		
6	Ethan "Tall Guy" Wilks		
7	Frank "The Ghost" Tomms		
8	Greg "Thriller" Dodson		
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

As shown in the initial dataset, the names follow a consistent pattern. By entering the double-quote

extraction formula into cell **B2**, we initiate the **parsing** process. The formula identifies the start of the nickname and the end, effectively ignoring the first name and surname. This creates a clean list of nicknames that can be used for further analysis or reporting purposes without manually re-typing each entry.

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

After entering the formula, the **Excel** "fill handle" can be used to propagate the logic down the entire column. This automation is a cornerstone of efficient **spreadsheet** management. As the formula is dragged down, the cell reference updates relatively, allowing the same logic to be applied to every athlete in the list regardless of how long their name or nickname might be.



The final result displays a perfectly isolated column of nicknames. This clean separation of data is essential for maintaining a high-quality **dataset**. Once extracted, these values can be copied and pasted as values to remove the underlying formulas, or they can remain dynamic so that any changes made to the original names in column A are immediately reflected in the extracted nicknames in column B.

Example 2: Practical Application for Single Quotes

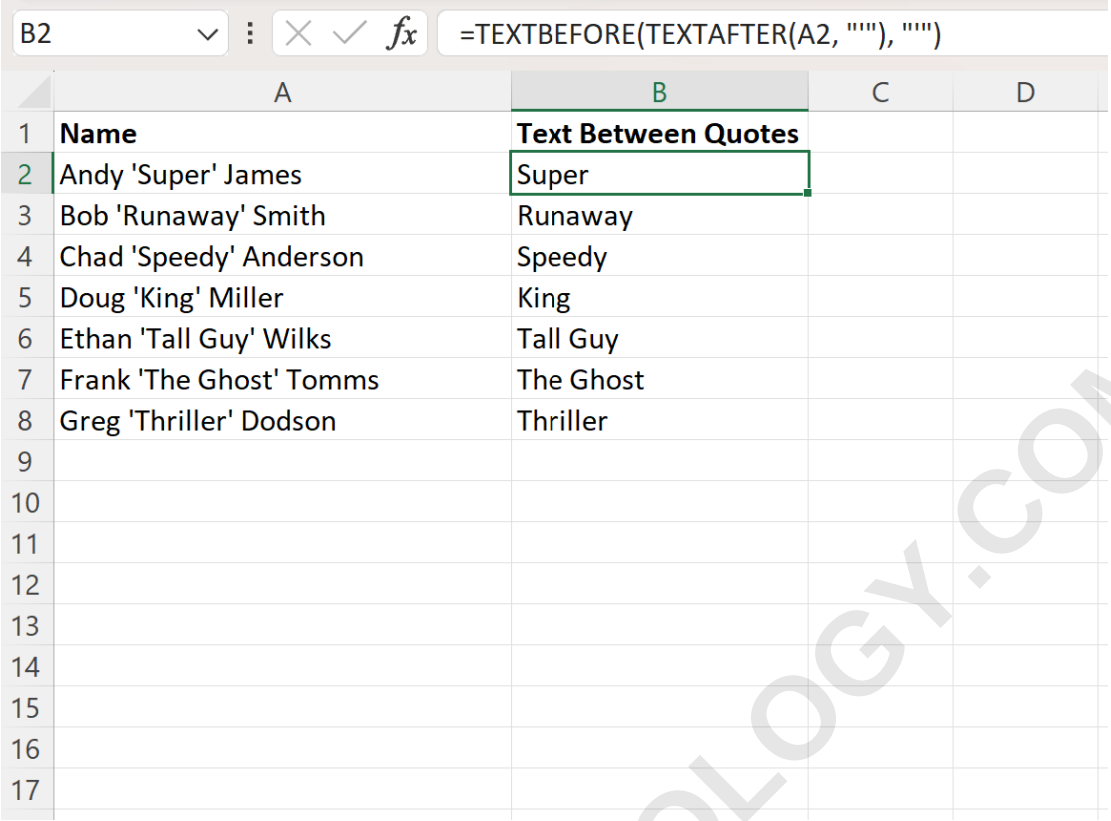
In a similar vein, datasets may arrive with single quotes instead of double quotes. This is common in various **metadata** formats or when data has been exported from systems that use single quotes to denote strings. The logic remains the same: we need to isolate the content between the first and second occurrence of the single quote character to ensure our **data cleaning** objectives are met.

	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				

In this second example, the athlete nicknames are surrounded by single quotes. While visually different, the structural challenge is identical. We apply the specific single-quote variation of our formula in the first cell of the output column. This ensures that the **Excel** engine looks for the correct **character** code during the extraction process, avoiding errors that would occur if we searched for double quotes in a single-quote field.

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

By dragging the formula down, we see the immediate benefits of **automation**. Each cell is processed individually, and the specific text within the single quotes is retrieved. This method is particularly robust because it does not rely on the position of the quotes within the string; whether the nickname is at the beginning, middle, or end of the cell, the **TEXTAFTER** and **TEXTBEFORE** combination will find it.



The resulting column is now ready for use in **Pivot Tables**, charts, or as part of a larger **VLOOKUP** or **XLOOKUP** operation. By isolating the quote-enclosed text, we have transformed a messy string into a discrete **data point**. This level of granularity is what allows for sophisticated business intelligence and deeper insights into the underlying information.

Advanced Considerations: Handling Errors and Multiple Quotes

While the nested **TEXTBEFORE** and **TEXTAFTER** approach is highly effective, it is important to consider scenarios where data might be inconsistent. For instance, if a cell is missing one or both quotes, the formula will return a **#N/A** error. To maintain a professional-looking **spreadsheet**, you can wrap your extraction formula in an **IFERROR** function. This allows you to specify a fallback value, such as a blank cell or a "Not Found" message, ensuring your reports remain clean and readable.

Furthermore, if a cell contains multiple sets of quotes, these functions provide additional arguments to specify which instance you wish to extract. By default, they look for the first instance. However, by adding an instance number to the **TEXTAFTER** or **TEXTBEFORE** parameters, you can precisely target the second, third, or even the last set of quotes in a long string. This level of control is essential for complex **data mining** tasks.

Another consideration is the presence of leading or trailing spaces within the quotes. Even after extraction, you might find that the nickname has a space before or after it. In such cases, wrapping the entire formula in the **TRIM** function is a best practice. This ensures that any "invisible" characters are removed, leaving only the text. Combining **TRIM** with our extraction formula creates a powerful tool for maintaining **data quality** across large-scale projects.

Comparison with Legacy Methods: MID and FIND

Before the introduction of these specific text functions in **Microsoft 365**, **Excel** users had to rely on a much more cumbersome combination of **MID**, **FIND**, and **LEN**. The legacy formula for extracting text between quotes often looked like a long string of nested logic that was difficult to read and even harder for others to understand. This older method required calculating the starting position of the first quote and subtracting it from the position of the second quote to determine the length of the string to be extracted.

The transition to **TEXTBEFORE** and **TEXTAFTER** represents a shift toward more human-readable formulas. In **software engineering**, readability is a key metric for code quality, and the same principle applies to complex spreadsheets. By using functions that explicitly state their purpose--getting text before or after a delimiter--the formula becomes self-documenting. This reduces the time needed for colleagues or future versions of yourself to audit the **spreadsheet** and understand how the data is being transformed.

Moreover, the modern functions are generally more performant. Legacy formulas involving multiple **FIND** operations can become "heavy" as the **spreadsheet** grows, leading to slower calculation times. The newer engine is optimized for these types of **string** operations, ensuring that even with tens of thousands of rows, your workbook remains responsive. This performance boost is a critical factor for organizations dealing with **Big Data** within a desktop environment.

Best Practices for Data Hygiene in Excel

Extracting text between quotes is often just one step in a broader **data cleaning** strategy. To ensure the best results, always work on a copy of your original data or use **Power Query** for more advanced transformations. **Excel** offers several built-in tools that complement formula-based extraction, such as "Flash Fill," which can sometimes recognize the pattern you are trying to achieve and complete the task automatically without formulas.

When using formulas, it is wise to keep your **source data** in its original format and place your extraction formulas in adjacent columns. This "non-destructive" editing style allows you to verify the results against the original input at any time. If you eventually need to move the extracted data to another system, remember to use "Paste Special" as values to break the link to the formula, which prevents errors if the source data is ever moved or deleted.

Finally, consider the **encoding** of your quotes. In some cases, text imported from word processors might use "smart quotes" or "curly quotes" ("" or '') instead of standard "straight quotes" (" or '). If your formula fails, check the character type. You may need to adjust your delimiter in the formula or use the **SUBSTITUTE** function to convert all curly quotes to straight quotes before performing the extraction.

Conclusion and Further Learning

Mastering the extraction of text between quotes is a significant milestone in becoming an **Excel** expert. By moving beyond basic data entry and into the realm of automated **string manipulation**, you significantly increase your value as an analyst. The **TEXTBEFORE** and **TEXTAFTER** functions are versatile tools that, once understood, can be applied to a vast array of challenges, from parsing **URL** parameters to cleaning up complex log files.

As you continue to refine your skills, explore other dynamic array functions such as **CHOOSECOLS**, **VSTACK**, and **LET**. These functions can be combined with our extraction techniques to create even more powerful data processing workflows. For instance, the **LET** function allows you to define variables within your formula, making complex nested extractions even easier to read and much faster to calculate.

To deepen your understanding of these topics, the following tutorials explain how to perform other common tasks in **Excel**, providing a pathway to complete mastery of the world's most popular data tool. By staying current with the latest updates to **Microsoft Office**, you ensure that your technical skills remain sharp and your data remains accurate, insightful, and ready for action.