How to Easily Find Text in Excel Using SEARCH and FIND

Authored by

stats writer

February 20, 2026

What is the difference between the SEARCH and FIND functions in Excel?

In the expansive realm of **data analysis** and management, **Microsoft Excel** remains an indispensable tool for professionals across various industries. One of the most fundamental yet critical tasks performed within this **spreadsheet** software is the identification and extraction of specific information from larger bodies of text. To facilitate this, Excel provides two primary tools: the **SEARCH** function and the **FIND** function. While they may appear identical at first glance--both designed to return the numerical position of a **substring** within a parent string--they possess distinct operational characteristics that dictate their suitability for different computational scenarios.

Understanding the nuances between these two functions is essential for maintaining data integrity and ensuring the accuracy of complex formulas. The primary differences lie in two specific areas: **case sensitivity** and the interpretation of **wildcard characters**. Choosing the incorrect function can lead to erroneous results, particularly when dealing with large datasets where manual verification is impossible. This guide aims to provide an exhaustive breakdown of how these functions behave, the logic behind their design, and practical examples to illustrate their application in real-world data environments.

As we delve deeper into the mechanics of these functions, it becomes clear that their utility is not just in finding a letter or a word, but in serving as the foundation for more advanced data manipulation. Whether you are cleaning a mailing list, parsing complex product codes, or automating report generation, mastering the **SEARCH** and **FIND** functions is a prerequisite for any advanced **Excel** user. By the end of this analysis, you will be able to discern which function to deploy based on the specific requirements of your data structure and the level of precision needed for your logical tests.

The Fundamentals of the SEARCH Function

The **SEARCH function** in Excel is designed for flexibility and ease of use, particularly when the exact casing of the target text is unknown or irrelevant. Its syntax, which consists of the search text, the text to be searched, and an optional starting position, allows users to locate the first occurrence of a specific character or string. Because it is not case-sensitive, a query for "apple" will successfully return the position of "Apple," "APPLE," or even "aPpLe." This makes it an ideal choice for processing user-generated content where capitalization may be inconsistent across different entries.

Beyond its case-insensitive nature, the most powerful feature of the **SEARCH** function is its native support for **wildcard characters**. These symbols, such as the asterisk (*) and the question mark

(?), allow users to perform pattern matching rather than just exact string matching. For instance, the question mark represents any single character, while the asterisk represents any sequence of characters. This functionality transforms the **SEARCH** function into a robust tool for identifying partial matches or variations within complex strings, such as finding a specific domain in an email address or a specific sequence within a serialized code.

In terms of output, the function returns an integer representing the starting position of the first occurrence of the found text. If the text is not located, the function returns a **#VALUE!** error. This behavior is crucial for error handling in nested formulas, where functions like **ISNUMBER** are often used in conjunction with **SEARCH** to return a boolean TRUE or FALSE value. This combination is a staple in conditional formatting and logical filtering, allowing analysts to categorize data based on the presence or absence of specific textual patterns without worrying about the specifics of capitalization.

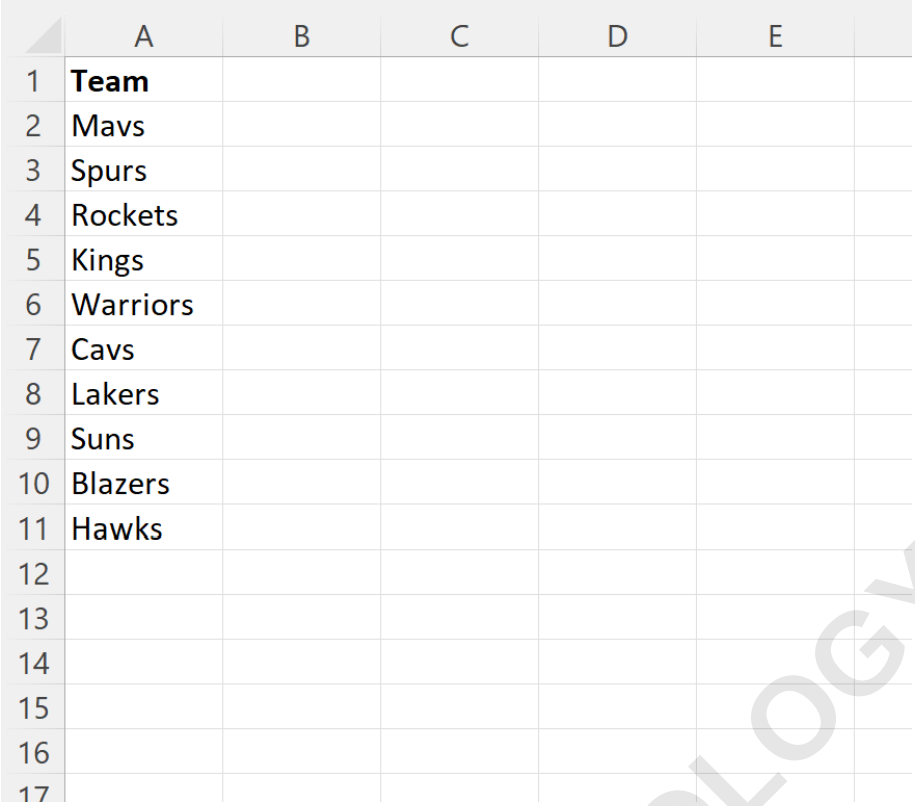
Analyzing the Rigidity of the FIND Function

Contrastingly, the **FIND function** is the tool of choice for scenarios requiring absolute precision and **case sensitivity**. Unlike its counterpart, **FIND** treats uppercase and lowercase letters as distinct entities. If you are searching for the string "ID" within a cell, **FIND** will ignore "id" or "Id," ensuring that your search is narrowed down to the exact match. This level of strictness is vital when working with technical identifiers, case-sensitive passwords, or specific **character encoding** where the case of a character conveys significant meaning.

Another defining characteristic of the **FIND** function is its lack of support for **wildcard characters**. When you input a question mark or an asterisk into the **FIND** function, it treats those symbols as literal characters rather than placeholders. This means that if you search for "test?", the function will only return a result if the literal string "test?" is present in the target cell. This lack of ambiguity is preferred in environments where data is highly structured and the presence of special characters must be identified literally rather than used as search logic.

The syntax for **FIND** is identical to that of **SEARCH**, requiring the text you want to find, the text to search within, and an optional start number to define where the search begins. While it may seem more restrictive, the **FIND** function's predictability is its greatest asset. It eliminates the risk of "false positives" that might occur with a case-insensitive search. For professional **data analysis**, this precision ensures that formulas behave exactly as intended, particularly when the data source is a controlled system output where case consistency is guaranteed.

The following image illustrates a typical dataset of basketball team names in **Microsoft Excel**, which we will use to demonstrate these functional differences:



	A	B	C	D	E
1	**Team**				
2	Mavs				
3	Spurs				
4	Rockets				
5	Kings				
6	Warriors				
7	Cavs				
8	Lakers				
9	Suns				
10	Blazers				
11	Hawks				
12					
13					
14					
15					
16					
17					

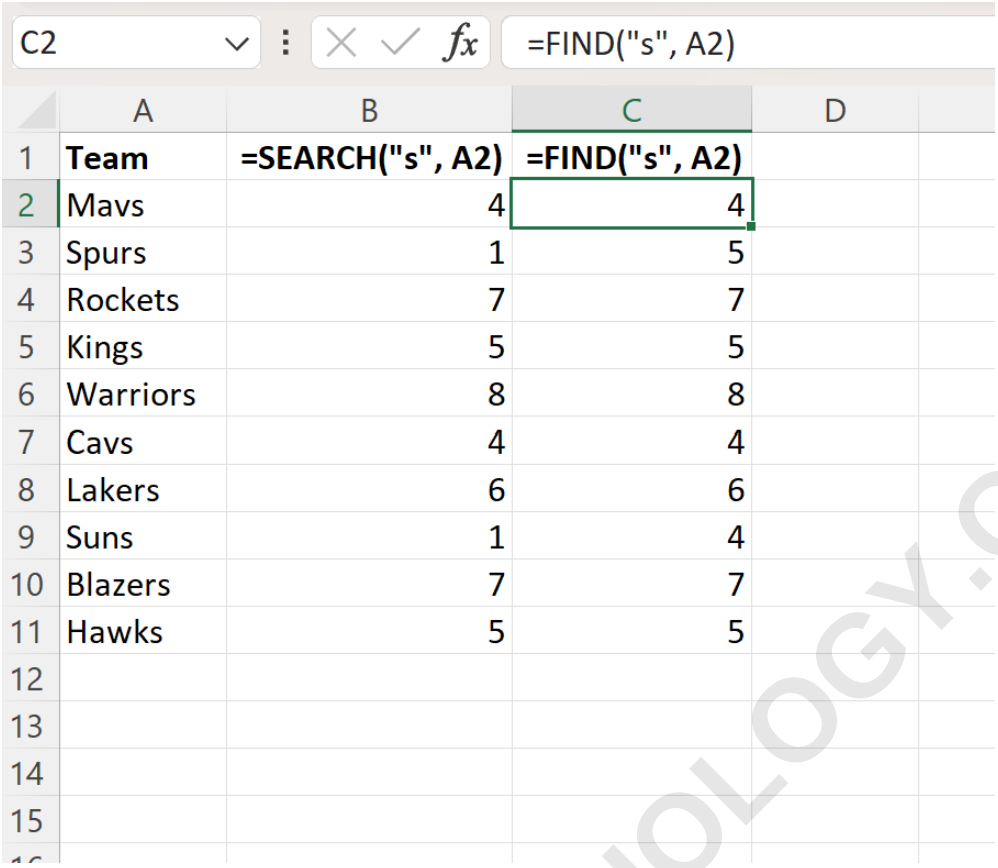
Example 1: Evaluating the Case-Sensitive Difference

To truly grasp the impact of **case sensitivity**, let us consider an example where we need to locate the position of the character "s" within our list of basketball teams. In this scenario, we apply both functions to the same set of data to observe how they interpret the character "s" versus the character "S." Because the team names are proper nouns, many begin with an uppercase "S," which provides the perfect testing ground for these two **substring** location methods.

By inputting the formulas into their respective columns, we can see a clear divergence in results. We will utilize the following formulas for our analysis in cells **B2** and **C2**:

B2: =SEARCH("s", A2)
C2: =FIND("s", A2)

After applying these formulas across the entire column, the resulting data reveals the fundamental behavioral difference between the two functions. The **SEARCH** function effectively ignores the case of the "s," while the **FIND** function adheres strictly to the lowercase requirement specified in the formula. This distinction is not merely academic; it determines whether a formula successfully identifies the start of a word or skips to a middle character, which can drastically alter subsequent data extraction steps.



	A	B	C	D
1	**Team**	**=SEARCH("s", A2)**	**=FIND("s", A2)**	
2	Mavs	4	4	
3	Spurs	1	5	
4	Rockets	7	7	
5	Kings	5	5	
6	Warriors	8	8	
7	Cavs	4	4	
8	Lakers	6	6	
9	Suns	1	4	
10	Blazers	7	7	
11	Hawks	5	5	
12				
13				
14				
15				

As observed in the visual results, the **SEARCH** function returns a value of **1** for the team "Spurs" because it identifies the uppercase "S" as a match for our search term "s." It treats the character as the same regardless of its visual or binary casing. This is the behavior most users expect when performing a general text search in a **spreadsheet** where the primary goal is to find a specific letter regardless of its position in a sentence or its capitalization.

In contrast, the **FIND** function returns a value of **5** for "Spurs." This is because the function bypasses the initial uppercase "S" at position 1 and continues searching until it encounters the first lowercase "s" at the end of the word. This behavior confirms that **FIND** is operating with strict **case sensitivity**. If the team name were "Suns," the **FIND** function would return 4, whereas **SEARCH** would return 1. This illustrates how **FIND** provides a more granular level of control for specific data formatting requirements.

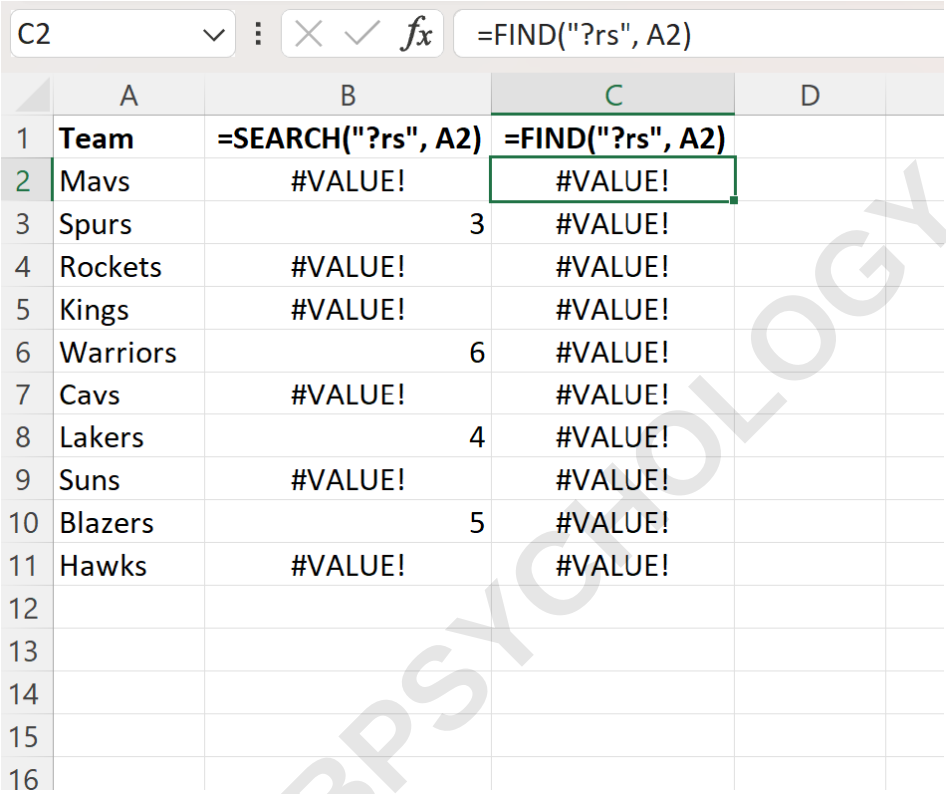
Example 2: Leveraging Wildcard Characters for Advanced Lookups

The second major differentiator is the support for **wildcard characters**. Wildcards are essential when you need to find a pattern rather than a specific, known string. In this example, we want to locate the position of a **substring** that ends in "rs," but we are indifferent to which character immediately precedes it. We use the question mark (?) wildcard, which serves as a placeholder for

any single character. This type of search is common when dealing with codes where the middle character might vary but the suffix remains constant.

We will apply the following pattern-matching formulas to our dataset to see how each function handles the special character "?" within the search string:

B2: =SEARCH("?rs", A2)
C2: =FIND("?rs", A2)



	A	B	C	D
1	**Team**	**=SEARCH("?rs", A2)**	**=FIND("?rs", A2)**	
2	Mavs	#VALUE!	#VALUE!	
3	Spurs	3	#VALUE!	
4	Rockets	#VALUE!	#VALUE!	
5	Kings	#VALUE!	#VALUE!	
6	Warriors	6	#VALUE!	
7	Cavs	#VALUE!	#VALUE!	
8	Lakers	4	#VALUE!	
9	Suns	#VALUE!	#VALUE!	
10	Blazers	5	#VALUE!	
11	Hawks	#VALUE!	#VALUE!	
12				
13				
14				
15				
16				

The results in column B demonstrate the power of the **SEARCH function**. By using the query "?rs", **SEARCH** looks for any three-character sequence where the second and third characters are "r" and "s." For the team "Lakers," it identifies "ker" as a mismatch but finds "ers" at the end of the string, returning the position where that three-character pattern begins. This flexibility allows for dynamic searching that can accommodate variations in data entry or complex naming conventions.

However, the **FIND function** fails completely in this scenario, returning the **#VALUE!** error for every entry. This occurs because **FIND** does not recognize the question mark as a wildcard. Instead, it literally searches for the string "?rs"--a question mark followed by the letters "r" and "s." Since none of our basketball team names contain a literal question mark, the function cannot find a match. This highlights why **SEARCH** is the superior choice for pattern recognition, while **FIND** should be reserved for literal string matches.

Functional Limitations and Error Management

Both **SEARCH** and **FIND** are susceptible to errors, the most common being the **#VALUE!** error. This error triggers when the specified **find_text** cannot be located within the **within_text** string. In a large **Microsoft Excel** workbook, these errors can propagate through dependent formulas, causing entire dashboards to fail. Therefore, it is standard practice to wrap these functions within an **IFERROR** or **ISNUMBER** statement. This ensures that instead of an error, the formula returns a zero, a blank, or a "Not Found" message, maintaining the visual and functional integrity of the **spreadsheet**.

Another limitation involves the optional **start_num** argument. If this argument is set to a value less than or equal to zero, or a value greater than the length of the string being searched, the function will again return a **#VALUE!** error. Precision in defining the starting point is necessary when you want to bypass a known occurrence of a string to find a subsequent one. For example, if you are looking for the second space in a full name, you would use the result of the first search (plus one) as the **start_num** for the second search.

It is also worth noting that neither function is capable of returning multiple positions simultaneously. They are designed to return only the index of the first occurrence found after the starting position. To find all occurrences of a character within a single cell, one would need to employ complex array formulas or **VBA** (Visual Basic for Applications) scripts. Understanding these boundaries allows users to design more effective logic and avoid over-complicating their **data analysis** workflows.

Optimizing Workflows: When to Use SEARCH vs. FIND

The decision to use **SEARCH** or **FIND** should be dictated by the nature of your data and the goal of your analysis. If you are dealing with "dirty" data--such as manually entered names, addresses, or survey responses--the **SEARCH** function is almost always the better choice. Its case-insensitivity acts as a safety net, ensuring that variations in capitalization do not result in missed data. Furthermore, if your search criteria involve patterns or partial strings, **SEARCH** is the only viable native function for the job.

Conversely, the **FIND** function should be utilized when you are working with structured, system-generated data where the case of the characters is significant. For instance, in many programming environments and **Linux**-based file systems, "File.txt" and "file.txt" are entirely different files. If your Excel workbook is interacting with or auditing such data, the case-sensitive precision of **FIND** is mandatory. It prevents the accidental grouping of distinct data points that happen to share the same letters but different casing.

In summary, the **SEARCH** function offers flexibility and is highly compatible with the unpredictable nature of human-entered text. The **FIND** function offers strict accuracy and is best suited for

technical data where capitalization is a controlled variable. By keeping these two distinctions--case sensitivity and wildcard support--at the forefront of your formula design, you can build more resilient and accurate **Excel** models that stand up to rigorous data validation requirements.

Conclusion and Further Learning

Mastering the **SEARCH** and **FIND** functions is a pivotal step in evolving from a basic user to an intermediate or advanced **Excel** analyst. These functions serve as the "eyes" of your formulas, allowing them to "see" and locate data within the textual noise of a cell. When combined with other text functions like **MID**, **LEFT**, and **RIGHT**, they enable sophisticated data parsing techniques that can save hours of manual labor and significantly reduce the likelihood of human error in **data analysis**.

As you continue to refine your skills, consider exploring how these functions interact with Excel's newer dynamic array features. While the core logic remains the same, the way these functions are deployed across thousands of rows is constantly evolving. Staying updated with the latest **official documentation** from Microsoft will ensure that you are utilizing these tools in the most efficient manner possible, leveraging the full power of the **spreadsheet** environment.

For those looking to expand their knowledge beyond text searching, there are numerous resources available that cover the breadth of Excel's capabilities. Whether you are interested in financial modeling, statistical analysis, or data visualization, the journey begins with a solid foundation in basic function logic. The following tutorials explain how to perform other common tasks in Excel and will further enhance your proficiency in managing complex data structures.