How can I use IF function with text values in Google Sheets?

Authored by

stats writer

November 18, 2025

Working with text data is one of the most common tasks performed in Google Sheets. While numerical analysis is straightforward, applying conditional logic based on text strings requires specific formula construction. The primary tool for this evaluation is the **IF function**, which allows you to test a condition and return one value if the condition is true and another if it is false. However, when dealing with text, especially partial matches or multiple criteria, the basic **IF** syntax must be combined with powerful secondary functions like SEARCH function, ISNUMBER function, and SUMPRODUCT function to achieve sophisticated results.

This comprehensive guide details the essential methods for using the IF function specifically with text values in Google Sheets. We will explore three distinct scenarios, ranging from simple equality checks to complex multiple-criteria comparisons, providing robust formulas and practical examples for each technique. Mastering these approaches will significantly enhance your ability to automate data classification and reporting based on textual properties, ensuring accurate and flexible data management within your spreadsheets.

The Foundation: Understanding Conditional Logic in Google Sheets

The IF function operates based on the principle of conditional logic: `=IF(logical_expression, value_if_true, value_if_false)`. When working with text, the `logical_expression` typically involves comparing a cell's content to a reference string. For simple checks, this comparison is straightforward using the equals sign (=). However, complex text analysis requires helper functions to transform the text comparison into a logical (Boolean) output that the **IF** function can interpret--a necessity for tasks like identifying substrings or handling variations in text structure which often influence the accuracy of the evaluation.

For instance, determining if a cell contains a specific word requires transforming the textual search into a numerical or true/false output. Functions like SEARCH function return a number indicating the position of the substring if found, or an error if not found. The resulting number or error must then be evaluated by another function, such as ISNUMBER function, which converts the result into a simple `TRUE` or `FALSE` output. This layered functional approach ensures that even the most nuanced textual evaluations can be boiled down to the simple binary choice required by the outer **IF** statement, allowing for powerful, automated data processing capabilities within your spreadsheet environment.

Method 1: Checking for Exact Text Equality

The most fundamental use case involves checking if the value of a specific cell is an exact match to a predefined text string. This method is crucial when dealing with standardized input, such as categorizing items based on precise labels or names, and relies solely on the equality operator (=) within the logical test of the IF function. It is important to note that, by default, text comparisons in

Google Sheets using the simple equals operator are generally **case-insensitive**. If strict, case-sensitive matching is required, you would need to incorporate the `EXACT` function, though for the majority of general comparison tasks, the standard equality check is perfectly suitable and preferred for its simplicity.

The structure of this formula is exceptionally clean and easy to implement. It directly compares the target cell against the specified text, which must be enclosed in double quotes (`" "`) to denote it as a text string rather than a numerical value or a cell reference. If the cell content matches the quoted string precisely, the formula returns the `value_if_true`; otherwise, it returns the `value_if_false`. This straightforward approach provides instant binary classification for your data records, making it ideal for tasks like flagging entries that deviate from a controlled vocabulary list.

You can use the following formula structure to check if a cell contains an exact match to a text value:

=IF(A2="Starting Center", "Yes", "No")

This logical operation evaluates the content of cell **A2**. If **A2** is identically equal to the text string "Starting Center", the formula will return the affirmative value, "Yes". If there are any differences--including leading or trailing spaces, or minor textual variations--it will return the negative value, "No." This forms the basis of simple data validation and categorization within Google Sheets.

Method 2: Searching for Specific Text Within a Cell (Partial Match)

Often, data fields contain complex descriptions or multiple attributes, necessitating a check for the presence of a specific keyword rather than an exact match of the entire cell content. To accomplish this partial matching, we must integrate the SEARCH function and the ISNUMBER function into our **IF** statement. The **SEARCH** function is the core engine that scans the text, locating the starting position of the keyword, and importantly, it is **case-insensitive** by default, making it highly versatile for general data clean-up and analysis where capitalization consistency cannot be guaranteed.

The mechanics work as follows: the **SEARCH** function attempts to find the specified text string (e.g., "Guard") within the target cell (**A2**). If successful, **SEARCH** returns a numeric value (its starting position, e.g., 1, 5, or 10). If the text is not found, **SEARCH** returns the `#VALUE!` error. Since the **IF** function requires a `TRUE` or `FALSE` statement, we wrap the **SEARCH** result inside the ISNUMBER function. **ISNUMBER** checks if the output of **SEARCH** is a number (meaning the text was found) and returns `TRUE`, or if it is an error (text not found) and returns `FALSE`. This Boolean outcome is precisely what the outer **IF** function needs to execute its conditional return values.

Use the following nested formula structure for conducting a powerful partial text search:

=IF(ISNUMBER(SEARCH("Guard", A2)), "Yes", "No")

If the result of the SEARCH function is a number (meaning the text "Guard" was successfully located anywhere within the value of cell **A2**), the ISNUMBER function evaluates to `TRUE`, causing the **IF** statement to return "Yes". Conversely, if the text is absent, **ISNUMBER** returns `FALSE`, and the formula yields "No." This methodology is indispensable for tagging data records based on keywords contained within descriptive fields, such as comments or complex product names.

Method 3: Handling Multiple Text Criteria (OR Logic)

A more advanced requirement is to check if a cell contains one of several possible text strings--a scenario often referred to as an "OR" condition for text matching. While the previous methods handle single criteria, checking for multiple possible keywords simultaneously demands the use of array processing functions, such as combining the SUMPRODUCT function with **SEARCH** and **ISNUMBER**. This complex, yet highly efficient, structure allows us to evaluate an array of text values against a single cell using a single, consolidated formula, avoiding the need for lengthy, nested OR statements.

In this sophisticated setup, the list of texts we are searching for (e.g., {"Backup", "Guard"}) is encapsulated within curly brackets (`{}`), creating an array constant. When **SEARCH** is applied to this array against cell **A2**, it returns an array of results (numbers or errors). We then use **ISNUMBER** to convert this array into an array of `TRUE`/`FALSE` values. The double negative (`--`) operator is crucial here: it coerces the `TRUE`/`FALSE` values into their numerical equivalents (1 for TRUE, 0 for FALSE). Finally, the SUMPRODUCT function sums these 1s and 0s. If the resulting sum is greater than zero, it means at least one of the keywords was found, fulfilling the "OR" criteria.

To implement this powerful multi-criteria OR logic, use the following formula. This structure is highly scalable; you can easily add more text values to the array constant inside the curly brackets to expand your search criteria without changing the fundamental formula logic:

=IF(SUMPRODUCT(--ISNUMBER(SEARCH({"Backup","Guard"},A2)))>0, "Yes", "No")

If the resulting sum from the SUMPRODUCT function is greater than 0, indicating that cell **A2** contains either "Backup" *or* "Guard" (or both), the IF function returns "Yes". If the count is 0, meaning neither keyword was found, it returns "No." This vectorized approach offers a clean and efficient way to handle complex grouping and filtering tasks based on diverse text attributes simultaneously.

Practical Application: Detailed Dataset Overview

To illustrate these three methodologies in a practical context, we will utilize a sample dataset focused on basketball player positions. This dataset includes various position descriptions, some being exact matches, some containing multiple roles, and others needing only partial keyword identification. We aim to populate a new column (Column C) with a simple binary evaluation ("Yes" or "No") based on the text found in the primary data column (Column A).

Observing the data below, Column A contains the full position description. Our goal is to derive logical conclusions from this text using the formulas discussed above. This hands-on approach will clearly demonstrate how each formula type handles variations in text structure, highlighting the strengths of exact match vs. partial match formulas. Before diving into the specific examples, review the dataset used for all subsequent demonstrations:

	A	B	C
1	**Position**	**Points**	
2	Backup Shooting Guard	12	
3	Starting Point Guard	15	
4	Starting Center	8	
5	Backup Center	4	
6	Starting Power Forward	23	
7	Backup Point Guard	9	
8	Starting Shooting Guard	34	
9	Backup Power Forward	14	
10	Starting Small Forward	20	
11	Backup Small Forward	7	
12			
13			
14			
15			
16			
17			

This dataset provides a perfect testing ground for comparing the strictness of equality checks versus the flexibility of substring searches. For instance, the first method will exclusively identify the exact text "Starting Center", while the second and third methods will flexibly handle terms embedded within longer strings, such as "Backup Guard" or "Starting Power Forward / Center", proving their utility in real-world data scenarios.

Example 1: Implementing Exact Match Logic

In our first demonstration, we apply Method 1 to identify only the rows where the position is exactly designated as "Starting Center". This strict comparison is critical for processes that depend on standardized data inputs and requires absolute conformity between the cell content and the search string. We will enter the formula into cell **C2** and then apply it down the entire column to quickly classify every record based on this stringent requirement.

We type the following specific formula into cell **C2**. This logic checks if the value in cell **A2** is precisely equal to "Starting Center", returning "Yes" only upon exact congruence and "No" otherwise. This formula is the simplest application of the **IF** function with text values:

=IF(A2="Starting Center", "Yes", "No")

After entering the formula in **C2**, we can swiftly apply this calculation to the remaining rows by clicking and dragging the fill handle down the column. This action automatically adjusts the cell reference (from **A2** to **A3**, **A4**, and so on) for each subsequent row, efficiently generating the classification results for the entire dataset without manual intervention:

C2 =IF(A2="Starting Center", "Yes", "No")

	A	B	C	D
1	**Position**	**Points**	**Starting Center?**	
2	Backup Shooting Guard	12	No	
3	Starting Point Guard	15	No	
4	Starting Center	8	Yes	
5	Backup Center	4	No	
6	Starting Power Forward	23	No	
7	Backup Point Guard	9	No	
8	Starting Shooting Guard	34	No	
9	Backup Power Forward	14	No	
10	Starting Small Forward	20	No	
11	Backup Small Forward	7	No	
12				
13				
14				
15				

As evident in the resulting Column C, only the rows where Column A contains the identical text "Starting Center" receive a "Yes" designation. Descriptions like "Starting Center / Power Forward" or the singular "Center" are correctly identified as mismatches and return "No," confirming the formula's strict adherence to the exact match criterion, which is vital when data precision is paramount.

Example 2: Implementing Partial Match Logic

Next, we move to a scenario where we need to identify any player designated as a "Guard," regardless of whether they are a "Starting Guard," "Backup Guard," or "Point Guard." This requires the use of the partial match logic, combining **IF**, ISNUMBER function, and SEARCH function, as detailed in Method 2. This approach offers much greater flexibility than the strict equality check, making it highly suitable for data fields containing unstructured or compound text, where keywords are embedded within longer strings.

We input the following formula into cell **C2**. This complex expression first searches for the substring "Guard" within cell **A2**. If found, **SEARCH** returns a position number, which ISNUMBER function converts to `TRUE`, triggering the "Yes" output from the **IF** statement. This confirms that the presence of the keyword, irrespective of its position or surrounding text, satisfies the condition:

=IF(ISNUMBER(SEARCH("Guard", A2)), "Yes", "No")

After inputting the formula, we again use the click-and-drag method to populate Column C entirely. This action iterates the partial matching test across all rows, enabling rapid categorization based on the presence of the keyword "Guard" anywhere within the text description in Column A:

C2 =IF(ISNUMBER(SEARCH("Guard", A2)), "Yes", "No")

	A	B	C
1	**Position**	**Points**	**Position Contains "Guard"**
2	Backup Shooting Guard	12	Yes
3	Starting Point Guard	15	Yes
4	Starting Center	8	No
5	Backup Center	4	No
6	Starting Power Forward	23	No
7	Backup Point Guard	9	Yes
8	Starting Shooting Guard	34	Yes
9	Backup Power Forward	14	No
10	Starting Small Forward	20	No
11	Backup Small Forward	7	No
12			
13			
14			
15			
16			
17			

The resulting Column C clearly demonstrates the power of the partial match formula: it returns "Yes" for every row that contains the string "Guard," including complex descriptions like "Starting Point Guard" and "Backup Guard," while correctly returning "No" for positions like "Center" or "Forward." This validates the utility of using the nested SEARCH function and **ISNUMBER** functions for flexible, substring-based textual analysis.

Example 3: Implementing Multiple Criteria Logic

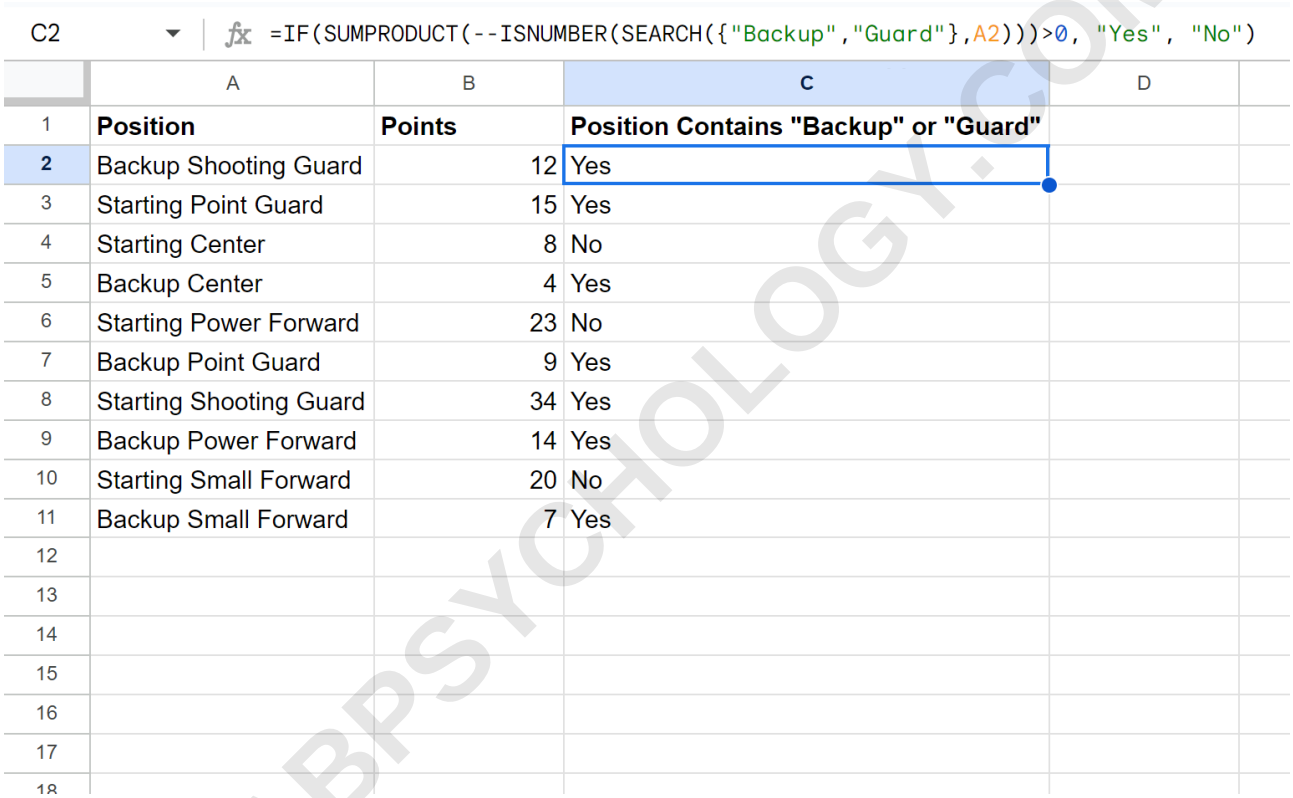
Finally, we tackle the scenario where we want to identify any player who is designated as either a "Backup" or a "Guard." This requires the robust array processing capability provided by Method 3, leveraging the SUMPRODUCT function in conjunction with the **IF** statement. This method is the ideal solution for efficiently checking multiple disjointed criteria within a single formula execution, simulating complex OR logic across an array of text strings.

We enter the complete array formula into cell **C2**. The array constant `{"Backup","Guard"}` ensures that the search is conducted against both terms simultaneously. The internal logic sums the count of findings; if the sum is greater than zero, it signifies that at least one of the conditions was met, satisfying the "OR" requirement and prompting a "Yes" response. This setup handles the multiple search terms much more gracefully than trying to chain multiple IF or OR statements

manually.

=IF(SUMPRODUCT(--ISNUMBER(SEARCH({"Backup","Guard"},A2)))>0, "Yes", "No")

By dragging the formula down column C, we apply this highly sophisticated array logic across the entire dataset. The dynamic nature of the formula ensures that we accurately tag any record that satisfies either the "Backup" condition or the "Guard" condition, providing a comprehensive assessment based on multiple text criteria simultaneously:



	A	B	C	D
1	**Position**	**Points**	**Position Contains "Backup" or "Guard"**	
2	Backup Shooting Guard	12	Yes	
3	Starting Point Guard	15	Yes	
4	Starting Center	8	No	
5	Backup Center	4	Yes	
6	Starting Power Forward	23	No	
7	Backup Point Guard	9	Yes	
8	Starting Shooting Guard	34	Yes	
9	Backup Power Forward	14	Yes	
10	Starting Small Forward	20	No	
11	Backup Small Forward	7	Yes	
12				
13				
14				
15				
16				
17				

The resulting classification shows a "Yes" for all rows containing either "Backup" (such as "Backup Center") or "Guard" (such as "Starting Shooting Guard"). This method is highly effective for complex categorization tasks, demonstrating how Google Sheets can execute powerful array operations on text data.

Note on Scalability: This formula structure is highly adaptable. Should your requirements grow, you can easily expand the scope of your search by including additional text values within the array constant (the curly brackets). For instance, `{"Backup","Guard", "Forward", "Starter"}` would check for four distinct terms simultaneously without altering the fundamental logic of the surrounding IF function and SUMPRODUCT function construction, providing a truly flexible solution.