How to Get a Yes/No Result for Matches in Google Sheets

Authored by

stats writer

January 16, 2026

In data analysis, efficiently determining the presence or absence of a specific value within a larger dataset is a common requirement. Utilizing a specialized Google Sheets formula allows users to execute powerful lookups that culminate in a straightforward, binary output: "Yes" or "No." This approach simplifies complex data validation, making it highly effective for flagging matches or non-matches based on precisely defined criteria. When the formula criteria are met--meaning the value is successfully located--it yields "Yes," confirming a match; conversely, if the conditions are not satisfied, it systematically returns "No," indicating an absence of the specified data point. This clear output mechanism is invaluable for subsequent data filtering, reporting, and automated decision-making processes.

The utility of a simple binary response cannot be overstated, particularly when dealing with large volumes of data where visual clarity is paramount. Instead of relying on potential error values like `#N/A` which are often returned by standard lookup functions, the "Yes" or "No" structure provides immediate interpretability. This method leverages sophisticated underlying functionality while presenting the result in a format easily understood by all stakeholders. Such conditional logic serves as a fundamental building block for creating highly responsive and automated spreadsheets, enabling users to rapidly assess compliance, inventory status, or membership verification without manually parsing complex results.

To achieve this precise binary determination, we combine three distinct and powerful functions: IF function, ISNUMBER function, and MATCH function. Each component plays a vital role in the overall execution. The MATCH function attempts the lookup, the ISNUMBER function validates the success of that attempt, and finally, the IF function translates that validation into the required textual output of "Yes" or "No." Understanding the synergy between these functions is essential for mastering conditional lookups in spreadsheet environments.

The Mechanics of the Yes/No Match Formula

The primary structure used to return a binary match indicator relies on nesting the verification functions within the core IF function. This powerful combination allows the formula to check if a specified value in one cell is present within a designated range elsewhere in the spreadsheet. If the presence is confirmed, the formula executes the true condition, returning "Yes"; otherwise, it executes the false condition, returning "No." This specific construction isolates the lookup logic entirely from the final user output, ensuring robust error handling.

The standard formula structure is defined as follows, using placeholders for the lookup value and the search range:

=IF(ISNUMBER(MATCH(C2,$A$2:$A$12,0)), "Yes", "No")

This specific formula is designed to evaluate whether the content of cell **C2** is found within the vertical array spanning from **A2** to **A12**. The result of this evaluation dictates the final output. If **C2** is successfully located, the formula confirms the match by returning **Yes**. Conversely, if the value is not located within the stipulated range, the formula decisively returns **No**. This architecture ensures absolute clarity regarding the membership status of the item being checked.

It is crucial to recognize that this formula addresses the shortcomings inherent in using the MATCH function alone for binary reporting. The MATCH function inherently returns either a numerical position (if successful) or an error (`#N/A`, if unsuccessful). By wrapping it with ISNUMBER and then IF, we transform a technical index or an error message into a user-friendly boolean text output. This layering of functionality is the core concept behind generating clean, conditional responses in Google Sheets.

Deconstructing the MATCH Function

The innermost function, MATCH function, is the engine of the entire lookup process. Its primary purpose is to search for a specified item within a range of cells and, if successful, return the relative positional index of that item within the range. It requires three critical arguments to operate effectively: the search key, the range to search, and the search type.

Let's break down the components as seen in the formula `MATCH(C2,$A$2:$A$12,0)`:

Search Key (`C2`): This is the value that the function is actively attempting to locate. In our example, this represents the individual team name we are checking for membership in the master list.

Range (`$A$2:$A$12`): This is the single-column or single-row array where the search key is expected to reside. It is essential that this range uses absolute references (denoted by the dollar signs, `$`) if the formula is intended to be dragged down across multiple rows, ensuring the lookup range remains fixed.

Search Type (`0`): This argument dictates how the match should be performed. Specifying `0` (or `FALSE`) forces an exact match. This is critically important for data validation tasks where partial or approximate matches are unacceptable. If an exact match is found, the function returns a number (e.g., 1, 5, 11, depending on the position); otherwise, it returns the standard `#N/A` error.

Understanding the output is key: a successful match results in a numerical value indicating position, while an unsuccessful match results in the `#N/A` error. This distinction is what the next function in our chain, ISNUMBER function, relies upon to perform its logical test. Because we are only interested in whether *any* match occurs, not the precise location, the numerical output itself merely serves as a confirmation signal.

The Role of ISNUMBER in Verification

Following the execution of the MATCH function, the result is fed directly into the ISNUMBER function. The purpose of ISNUMBER is simple yet powerful: it checks whether the value it receives is a numerical type and returns a boolean value--`TRUE` or `FALSE`--accordingly.

If the MATCH function finds the lookup value, it returns a number (the position index). ISNUMBER sees this number and evaluates to **TRUE**. Conversely, if the MATCH function fails to find the value, it returns the error value `#N/A`. Since `#N/A` is not a number, ISNUMBER evaluates to **FALSE**.

This crucial intermediary step effectively translates the technical output of the lookup operation (position index vs. error code) into the simple boolean input required by the outer IF function. It shields the overall formula from terminating due to the `#N/A` error and converts the complex lookup status into a clean, actionable logical test (TRUE/FALSE) that dictates the final "Yes" or "No" outcome.

Applying the IF Function for Binary Output

The outermost layer of this formula stack is the IF function, which provides the necessary structure to produce a custom, human-readable output based on the result of the logical test performed by ISNUMBER. The IF function requires three primary components: the logical expression, the value if true, and the value if false.

The logical expression is the entire `ISNUMBER(MATCH(...))` segment. As established, this expression resolves to either `TRUE` (match found) or `FALSE` (no match found).

Logical Expression: `ISNUMBER(MATCH(C2,$A$2:$A$12,0))`

Value if TRUE: `"Yes"` (The formula returns this text string when a match is confirmed.)

Value if FALSE: `"No"` (The formula returns this text string when no match is found.)

This structure completes the objective of translating a data validation check into a definitive binary answer, making the resulting spreadsheet far more intuitive for general analysis. This application of IF function demonstrates sophisticated conditional logic programming within a spreadsheet environment. It ensures that regardless of the complexity of the underlying lookup operation, the end-user always receives a clear and unambiguous response. This design principle is vital for creating reports that are immediately actionable and require minimal interpretation.

Step-by-Step Example: Implementing Match Checks

To illustrate the practical application of this conditional match formula, consider a scenario

involving two lists of data: a master list of all valid entries and a specific list of items that require validation against the master. Suppose we have a definitive list of basketball teams in column A and we need to verify if each team listed in column C belongs to that larger, authorized list in column A.

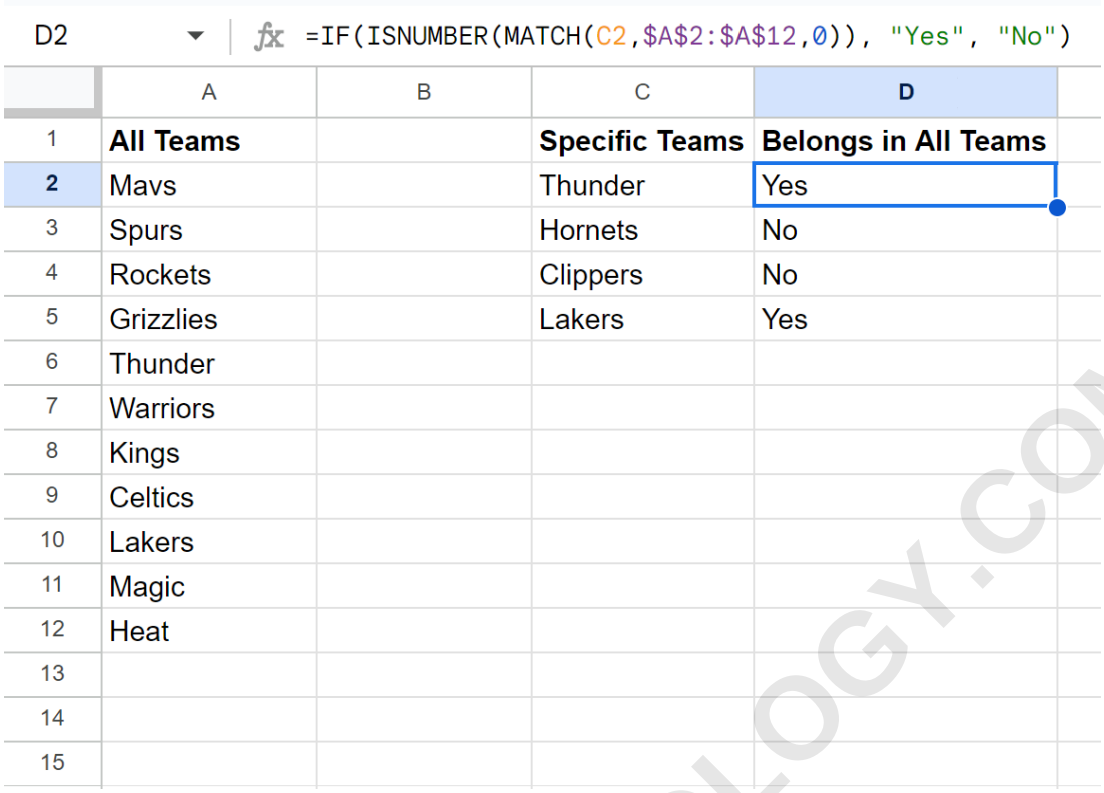
The visual representation below displays the initial dataset setup. Column A contains the comprehensive list ("All Teams"), and Column C contains the individual items we intend to check ("Specific Teams").

	A	B	C	D
1	**All Teams**		**Specific Teams**	
2	Mavs		Thunder	
3	Spurs		Hornets	
4	Rockets		Clippers	
5	Grizzlies		Lakers	
6	Thunder			
7	Warriors			
8	Kings			
9	Celtics			
10	Lakers			
11	Magic			
12	Heat			
13				
14				
15				
16				

To begin the verification process, we must enter the core formula into the first cell of our results column, which is typically **D2**, referencing the first item to be checked in **C2**. The formula requires the specific lookup value (C2) and the fixed range of the master list ($A$2:$A$12).

=IF(ISNUMBER(MATCH(C2,$A$2:$A$12,0)), "Yes", "No")

Once this formula is entered into cell D2, we can then apply standard spreadsheet practices by clicking and dragging the formula down to populate the remaining cells in column D.



	A	B	C	D
1	**All Teams**		**Specific Teams**	**Belongs in All Teams**
2	Mavs		Thunder	Yes
3	Spurs		Hornets	No
4	Rockets		Clippers	No
5	Grizzlies		Lakers	Yes
6	Thunder			
7	Warriors			
8	Kings			
9	Celtics			
10	Lakers			
11	Magic			
12	Heat			
13				
14				
15				

The resulting output in Column D provides an instant validation summary, clearly indicating whether each team in the "Specific Teams" list is present in the "All Teams" list. For example:

Thunder belongs in the All Teams list so the formula returns **Yes**.
Hornets does not belong in the All Teams list so the formula returns **No**.

This simple mechanism allows for immediate auditing of the specific teams list against the authorized master list.

Handling Ranges and Absolute References

A critical element of multi-cell formula application is the proper use of cell referencing, specifically the distinction between relative and absolute references. In the formula `=IF(ISNUMBER(MATCH(C2,$A$2:$A$12,0)), "Yes", "No")`, the lookup value **C2** uses a relative reference, while the lookup range **$A$2:$A$12** employs absolute references.

The use of a relative reference for **C2** is intentional. When the formula is copied or dragged down from D2 to D3, D4, and subsequent cells, the reference automatically adjusts to **C3**, **C4**, and so forth. This ensures that the formula always checks the value in the current row of the "Specific Teams" column against the master list.

Conversely, the use of absolute references--achieved by prepending a dollar sign ($) to both the

column letter and the row number, as in **$A$2:$A$12**--ensures that the range definition remains perfectly fixed, regardless of where the formula is copied. If the dollar signs were omitted (i.e., A2:A12), dragging the formula down would cause the range to shift (e.g., A3:A13, A4:A14), potentially leading to inaccurate results or failure to check the full master list. Utilizing absolute referencing is a fundamental best practice when defining fixed search ranges in spreadsheet lookups.

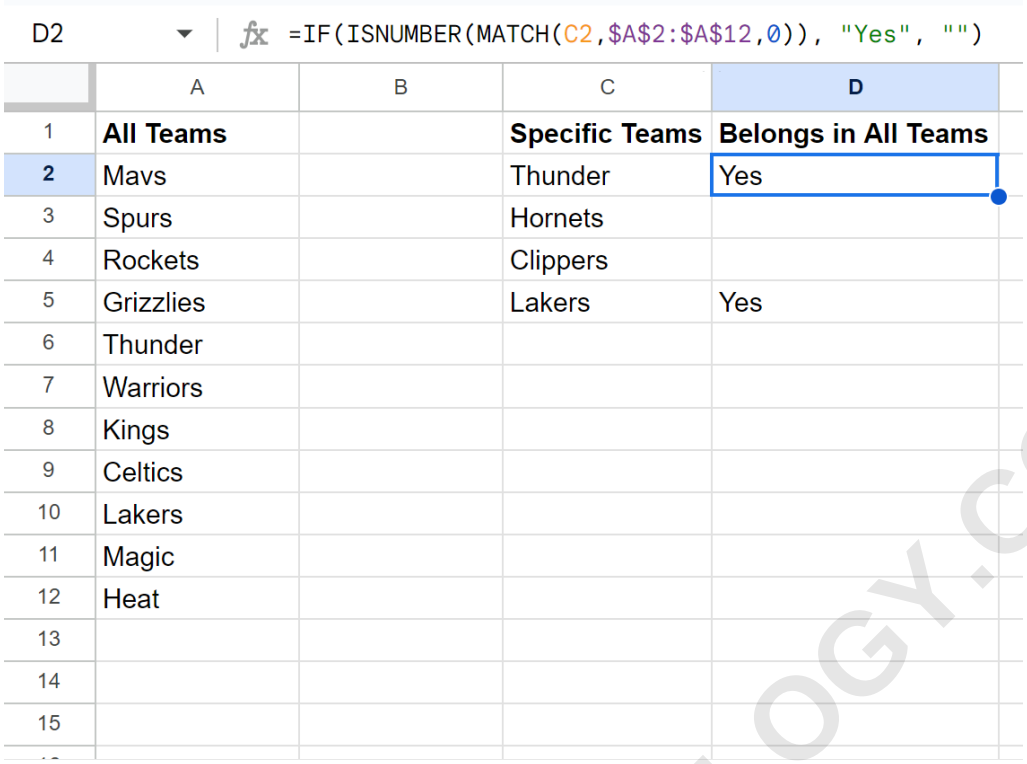
Alternative Output: Returning a Blank Value

While returning "Yes" or "No" offers maximum clarity, certain reporting requirements necessitate a cleaner output where only the confirmed matches are explicitly marked, and non-matches are left visually blank. This modification is simple to implement by altering the "value if false" argument of the encompassing IF function.

To instruct the formula to return an empty cell instead of the text "No," we replace the second quoted string with a pair of empty quotation marks (""). The modified formula structure is presented below:

=IF(ISNUMBER(MATCH(C2,$A$2:$A$12,0)), "Yes", "")

As shown in the screenshot below, if a team name in the **Specific Teams** list does not belong in the **All Teams** list, the formula now evaluates the FALSE condition and simply returns a blank value in the results column.



	A	B	C	D
1	**All Teams**		**Specific Teams**	**Belongs in All Teams**
2	Mavs		Thunder	Yes
3	Spurs		Hornets	
4	Rockets		Clippers	
5	Grizzlies		Lakers	Yes
6	Thunder			
7	Warriors			
8	Kings			
9	Celtics			
10	Lakers			
11	Magic			
12	Heat			
13				
14				
15				

This visual filtering technique emphasizes the positive confirmations and minimizes clutter when non-matches do not need explicit negative identification.

Advanced Considerations and Alternative Methods

While the combination of IF function, ISNUMBER function, and MATCH function is highly effective and widely compatible across spreadsheet versions, users working in modern Google Sheets may consider alternative functions that achieve similar conditional matching with slightly streamlined syntax.

For instance, functions such as `COUNTIF` or `XLOOKUP` can also be integrated into an IF structure to perform existence checks. Using `COUNTIF` offers a slightly more direct approach for checking membership, as it naturally returns a number greater than zero if a match is found, eliminating the need for the intermediate ISNUMBER function. The equivalent formula using `COUNTIF` would be: `=IF(COUNTIF($A$2:$A$12, C2) > 0, "Yes", "No")`. This alternative achieves the same result by testing if the count of the matching item is greater than zero, relying on core conditional logic.

Furthermore, the introduction of dynamic array functions and modern lookups like `FILTER` or `XLOOKUP` provides even greater flexibility, especially when dealing with complex multi-criteria matching. However, for the specific task of generating a simple, robust "Yes" or "No" based on a single-column match, the `IF(ISNUMBER(MATCH(...)))` structure remains an essential and highly reliable technique that every data analyst should master. Mastering this nested structure provides

a foundational understanding of how to manage error values and convert technical results into clear, actionable reports.