Learn How to Check if a Google Sheets Cell Contains Text from a List

Authored by

Mohammed looti

February 22, 2026

Introduction to Dynamic Text Matching in Google Sheets

Efficiently managing and analyzing substantial volumes of data within Google Sheets frequently requires sophisticated methods for checking text values. Specifically, analysts often need to determine if a target cell contains any of the keywords defined in a predefined list. This capability is fundamental when performing critical tasks such as categorizing records based on multiple criteria, executing complex data filtering operations, or triggering conditional logic based on the presence of specific keywords.

While simple string functions might suffice for checking a single keyword, scaling this operation to check against an entire dynamic list requires a more robust and efficient solution. Standard methods often involve cumbersome nested formulas or require manual dragging, which are prone to errors and lack scalability. Our objective is to introduce a single, powerful formula capable of processing an entire column instantly, maintaining high accuracy and flexibility across varying data sizes.

This comprehensive guide details a powerful method that leverages advanced Google Sheets functions to achieve this dynamic text-matching capability. The core approach relies on orchestrating a combination of specialized functions--namely ARRAYFORMULA, IF, LEN, REGEXMATCH, and TEXTJOIN--to create a mechanism that efficiently searches for list items within a data column, regardless of their position or case.

Understanding the Master Formula for Array Processing

To execute a comprehensive check on whether a cell contains text from a list in Google Sheets, we employ a single, elegant formula. This formula is designed for non-array input ranges, providing a resulting output column that reports TRUE if a match is found and FALSE otherwise, eliminating the need for copy-pasting the function down thousands of rows. The efficiency gain provided by this array processing is significant for large-scale data operations.

=ArrayFormula(IF(LEN(A2:A13), REGEXMATCH(A2:A13,".*(?i)("&TEXTJOIN("|", TRUE,$E$2:$E$4)&").*") ,""))

This master formula is structured in three logical parts: first, the ARRAYFORMULA wrapper enables vertical output expansion; second, the IF/LEN structure handles input cleanliness by ignoring empty cells; and finally, the core logic executed by REGEXMATCH performs the actual list lookup. Understanding how these components interact is key to customizing the formula for different scenarios and ensuring optimal performance.

A Detailed Breakdown of Core Formula Components

The strength of this solution lies in the precise combination of functions, each performing a specialized task necessary for handling array inputs and complex pattern matching:

ARRAYFORMULA: This foundational wrapper is responsible for applying the formula to every cell within the specified input range (e.g., A2:A13) and returning the results to adjacent cells automatically. This eliminates the tedious process of manual formula replication, significantly speeding up workflow and ensuring consistency across the entire output column.
IF(LEN(A2:A13), ..., ""): This segment acts as a vital error-prevention mechanism. The LEN function calculates the character length of the content in each cell in the input range. If the length is zero (i.e., the cell is blank), the IF statement returns an empty string (""). This prevents blank rows from incorrectly being evaluated by REGEXMATCH, ensuring that the final output is clean and only contains results for actual data entries.
REGEXMATCH(A2:A13, ".*(?i)("&TEXTJOIN("|", TRUE, $E$2:$E$4)&").*"): This is the functional core, using regular expression matching to find keywords.

TEXTJOIN("|", TRUE, $E$2:$E$4) dynamically creates a single search string from your lookup list. The pipe symbol ("|") is used as the delimiter because it functions as the logical "OR" operator in regular expressions, ensuring that the formula returns TRUE if *any* term in the list is found. The TRUE argument instructs the function to ignore any empty cells within the lookup range.
The full regular expression pattern is constructed as: <code>".*(?i)("&&").*"</code>.

<code>.*</code> at the start and end acts as a wildcard, allowing the keyword to appear anywhere within the target cell's content.
<code>(?i)</code> is a regular expression flag that enforces case-insensitivity, meaning the search will match "Mavs," "mavs," or "MAVS" equally.
The parentheses <code>(...)</code> group the output of the TEXTJOIN function, ensuring the "OR" operation is applied correctly to the entire list of keywords.

In summation, this complex formula dynamically constructs a precise, case-insensitive regular expression that determines whether any text in your specified input range (A2:A13) contains any of the values from your lookup list ($E$2:$E$4), yielding a TRUE or FALSE result.

Practical Application: Setting Up the Scenario Data

To fully grasp the utility of this formula, let us apply it to a practical data analysis problem. Imagine you are managing a large dataset detailing athlete performance, including player names, associated teams, and various statistics. Your specific task is to quickly flag all records related to teams based in the state of Texas, requiring a comparison of each team name against a fixed list of Texas-based organizations.

Our goal is to analyze the contents of the "Team" column and verify if the text within each cell matches any entry from our specific lookup list of teams. This targeted identification process is crucial for subsequent filtering, statistical analysis, or reporting based on geographic criteria.

Below is the initial structure of our dataset, where Column A contains the team names we need to scrutinize:

	A	B	C	D
1	**Team**	**Points**		
2	Mavs	31		
3	Nets	24		
4	Mavs	23		
5	Lakers	23		
6	Warriors	20		
7	Thunder	19		
8	Spurs	15		
9	Mavs	19		
10	Spurs	23		
11	Rockets	40		
12	Hornets	37		
13	Spurs	20		
14				
15				
16				
17				
18				
19				
20				

From the initial data, we must create a corresponding output column that flags whether the team belongs to one of the three major Texas NBA franchises: the Mavs, Spurs, or Rockets. This systematic flagging process will allow us to instantly isolate the desired records without manual cross-referencing.

Step-by-Step Implementation Guide

Implementing the dynamic text-matching solution is a straightforward, two-step procedure designed to separate your lookup criteria from your main data column, ensuring modularity and easy maintenance:

Define Your Keyword List: The first essential step is creating and defining a dedicated list of the keywords you intend to search for. For our example, this list comprises the shorthand names of the Texas NBA teams. It is best practice to place this list in a separate, easily identifiable column--Column E, for instance--and to ensure the range is referenced absolutely (using dollar signs, e.g., $E$2:$E$4) within the formula to prevent errors if the formula or data is moved.

	A	B	C	D	E
1	**Team**	**Points**			**List**
2	Mavs	31			Mavs
3	Nets	24			Spurs
4	Mavs	23			Rockets
5	Lakers	23			
6	Warriors	20			
7	Thunder	19			
8	Spurs	15			
9	Mavs	19			
10	Spurs	23			
11	Rockets	40			
12	Hornets	37			
13	Spurs	20			
14					
15					
16					
17					
18					
19					
20					

Apply the Array Formula to Your Data: Once the lookup list is established, insert the complete formula into the first cell of your desired output column (e.g., cell **C2**). This single entry point leverages the power of ARRAYFORMULA to automatically execute the check against every entry in the input column and populate the results vertically down the sheet.

=ArrayFormula(IF(LEN(A2:A13), REGEXMATCH(A2:A13,".*(?i)("&TEXTJOIN("|", TRUE,$E$2:$E$4)&").*") ,""))

Ensure that the formula references are correctly mapped: the input range (A2:A13) should cover all your data rows, and the lookup range ($E$2:$E$4) must be an absolute reference to your list of Texas teams. Since this is an ARRAYFORMULA, you only need to input it once in Google Sheets; do not attempt to drag the corner handle down.

C2 =ArrayFormula(IF(LEN(A2:A13), REGEXMATCH(A2:A13,".*(?i)("&TEXTJOIN

	A	B	C	D	E	F
1	**Team**	**Points**	**Texas Team?**		**List**	
2	Mavs	31	TRUE		Mavs	
3	Nets	24	FALSE		Spurs	
4	Mavs	23	TRUE		Rockets	
5	Lakers	23	FALSE			
6	Warriors	20	FALSE			
7	Thunder	19	FALSE			
8	Spurs	15	TRUE			
9	Mavs	19	TRUE			
10	Spurs	23	TRUE			
11	Rockets	40	TRUE			
12	Hornets	37	FALSE			
13	Spurs	20	TRUE			
14						
15						
16						
17						
18						
19						
20						

Analyzing and Verifying the Output

The successful execution of the formula yields immediate, actionable results in your designated output column (Column C). The output provides a clear indication--either TRUE or FALSE--of whether the corresponding team name contains one of the targeted keywords. A result of TRUE confirms that the team name includes a string match from the lookup list, while FALSE indicates the absence of any matching keyword.

By reviewing the results, we can verify the accuracy and case-insensitivity of the REGEXMATCH logic. Any row associated with "Mavs", "Spurs", or "Rockets" will accurately return **TRUE**, irrespective of capitalization. This rapid assessment capability is invaluable when dealing with large dataset verification.

Let's examine specific entries from our output to confirm the formula's precise handling of different input values:

The first record, featuring "Mavs", correctly generated a value of **TRUE**, affirming its presence in our Texas teams list.
The row containing "Nets" resulted in **FALSE**, as this team name is not included in our predefined

lookup criteria.
A subsequent entry for "Mavs" was also appropriately marked as **TRUE**, demonstrating the consistent application of the array formula.
The record for "Lakers" returned **FALSE**, as it does not contain any of the keywords defined in our Texas team list.

This systematic, array-based output provides a powerful foundation for subsequent data analysis, allowing for straightforward segmentation, filtering, and reporting based on the presence of specific keywords.

Enhancing Data Workflow with Advanced Techniques

Mastering this dynamic, list-based keyword matching technique represents a significant advancement in your Google Sheets proficiency. The ability to check against multiple criteria simultaneously using a single formula dramatically improves efficiency compared to relying on manual checks or complex nested IF/OR statements.

For advanced users, this core formula can be integrated into even more complex data management operations. Consider nesting this ARRAYFORMULA result within functions like QUERY or FILTER to create dynamic reports that update instantly as source data or the lookup list changes. Furthermore, the TRUE/FALSE output is perfectly suited for applying conditional formatting rules, visually highlighting matching records across large spreadsheets.

Continuing to explore the extensive capabilities of Google Sheets, particularly its array processing and regular expression functions, will enable you to tackle increasingly complex data challenges with greater speed and accuracy.

The following resources offer further guidance on various functions and techniques, helping you to enhance your spreadsheet proficiency and optimize your data analysis workflow: