How to Find Matching Values Between Two Columns in Google Sheets

Authored by

stats writer

November 19, 2025

In the dynamic environment of data analysis, particularly within popular spreadsheet applications like Google Sheets, the requirement to perform robust cross-column comparisons is a frequent necessity. While simple functions like **COUNTIF** can offer basic quantification--telling you *how many times* a value appears--a more direct and powerful approach is often needed to confirm binary existence: determining simply *if* a value from one list is present within another list. This critical function is essential for inventory management, data auditing, reconciliation, and validation tasks where speed and absolute accuracy are paramount. By leveraging a complex yet efficient combination of logical and lookup functions, users can achieve a clean, Boolean (**TRUE**/**FALSE**) result for every item in their data set, eliminating manual checks and ensuring systematic integrity across large volumes of data.

The Foundation: Combining MATCH, ISERROR, and NOT Functions

To achieve a reliable and scalable existence check in Google Sheets, we utilize a powerful nested formula that integrates the **MATCH**, **ISERROR**, and **NOT** functions. This combination systematically performs a lookup, determines if the lookup failed, and then inverts the result to provide the desired existence confirmation. This method is preferred for its precision and efficiency, as it immediately returns a result once a match is identified, unlike some alternative array-based solutions which may process the entire range unnecessarily.

The core objective is to convert the potential error output of the lookup operation into a meaningful logical value. If an item is found, the **MATCH** function returns a numeric position. If it is not found, **MATCH** returns an `#N/A` error. The ISERROR function captures this error status, returning **TRUE** only when the `#N/A` error occurs (i.e., when no match is found). Finally, the NOT function inverts this result, ensuring that if a match *is* found, the formula returns **TRUE**, meeting our objective of confirming existence.

The standardized formula syntax requires meticulous attention to cell referencing, particularly the use of absolute references (`$`) for the lookup range. This guarantees that when the formula is copied down multiple rows, the search column remains fixed while the cell being evaluated dynamically updates. This ensures the integrity and consistency of the lookups across the entire dataset.

The following structured formula provides the cleanest way to check if a specific column value exists within a defined range in Google Sheets, returning a definitive Boolean result:

=NOT(ISERROR(MATCH(A2,$B$2:$B$16,0)))

This specific construction executes a precise lookup: it checks if the value residing in cell **A2** successfully exists within the static range defined as **$B$2:$B$16**.

If the value from Column A is successfully located within the reference range **B2:B16**, the entire formula evaluates to **TRUE**. Conversely, if no corresponding entry is found, the formula accurately returns **FALSE**, signaling the item's absence.

Understanding the role of each component is essential for troubleshooting and modification. The MATCH function looks up `A2` in the fixed array `$B$2:$B$16`. If it fails, ISERROR function catches the resulting `#N/A`, yielding **TRUE**. The outermost NOT function then flips this **TRUE** (meaning "Error occurred, no match") to **FALSE**, providing the final, logical output expected for a failed lookup.

Illustrative Example: Auditing Inventory Stock

To demonstrate this technique in a practical business context, let us consider a scenario involving inventory reconciliation. Imagine we have two columns: a comprehensive **Grocery List** detailing all items needed for the week (Column A) and a current **Grocery Inventory** list showing what is presently available in stock (Column B). Our goal is to quickly generate a third column that reports, for every item on the needed list, whether or not it is currently in stock. This determination must be instantaneous and error-free.

The dataset below presents this situation clearly. We need to evaluate each item in Column A against the entire list in Column B. This is a common requirement in logistics, compliance, and large-scale data merging operations where confirming membership in a set is the core requirement.

	A	B	C	D
1	**Grocery List**	**Grocery Inventory**		
2	Apples	Avocado		
3	Bananas	Peaches		
4	Carrots	Bananas		
5	Pears	Apples		
6	Peppers	Watermelon		
7		Flour		
8		Bread		
9		Chips		
10		Milk		
11		Yogurt		
12		Cheese		
13		Pears		
14		Celery		
15		Cliantro		
16		Potatoes		
17				
18				
19				

We are specifically tasked with checking if each item listed in the **Grocery List** column (A2:A16) is also successfully present within the **Grocery Inventory** column (B2:B16). The absolute referencing structure is non-negotiable here, ensuring that when we evaluate 'Apples' (A2), 'Bananas' (A3), and so on, they are all measured against the identical, fixed inventory list.

Applying and Propagating the Existence Formula

To begin the reconciliation process, the formula is first entered into the header row of our resulting column, specifically cell **C2**. This initial step calculates the status for the first item on the list. The critical element here is verifying that the absolute reference (`$B$2:$B$16`) is correctly implemented before propagation.

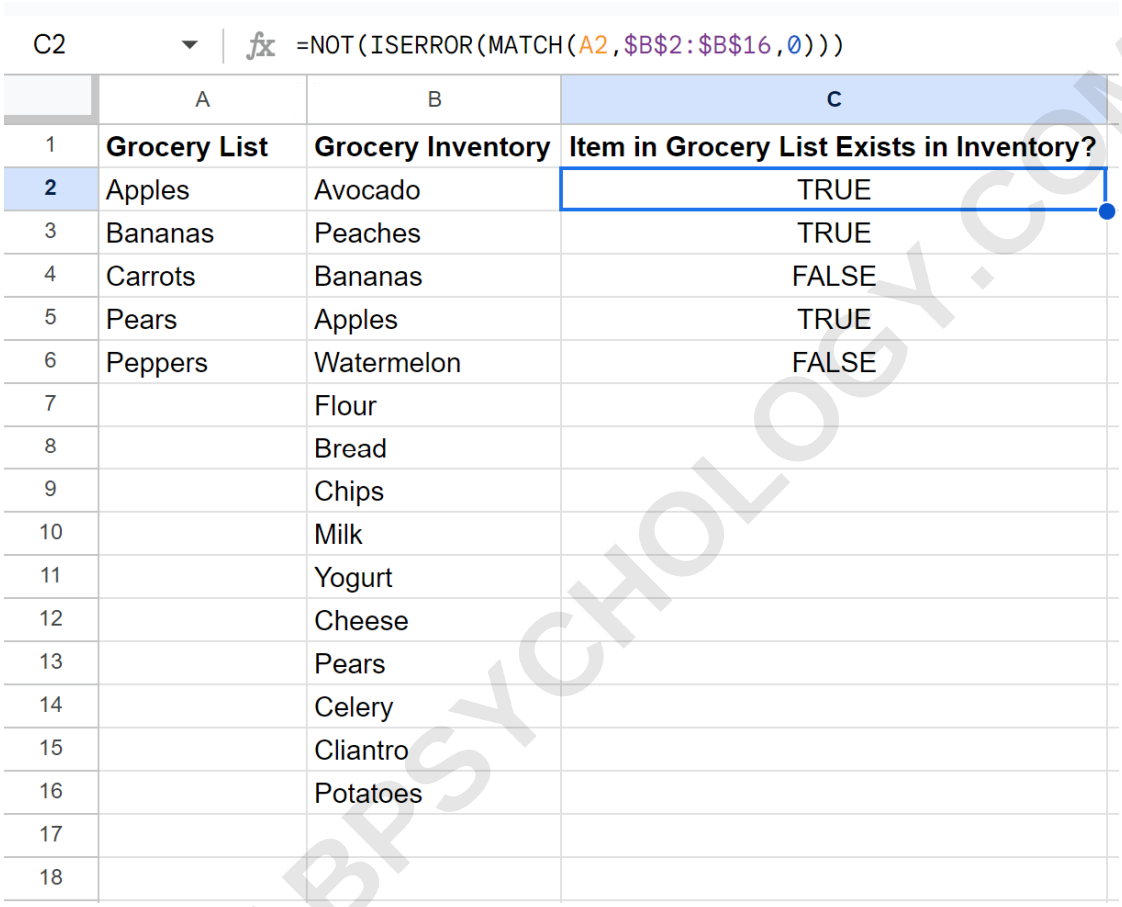
We apply the following formula directly to cell **C2**, targeting the first lookup value (A2):

=NOT(ISERROR(MATCH(A2,$B$2:$B$16,0)))

Once the formula is entered, the next step involves using the fill handle to drag the formula down the entire length of Column C, covering all corresponding rows in Column A. This action

automatically updates the relative cell reference (A2 becomes A3, A4, etc.) while the fixed inventory range remains securely locked. This propagation efficiently calculates the existence status for every single item without requiring manual input for each row.

We can then drag and fill this formula down to each remaining cell in column C, resulting in the following output:



	A	B	C
1	**Grocery List**	**Grocery Inventory**	**Item in Grocery List Exists in Inventory?**
2	Apples	Avocado	TRUE
3	Bananas	Peaches	TRUE
4	Carrots	Bananas	FALSE
5	Pears	Apples	TRUE
6	Peppers	Watermelon	FALSE
7		Flour	
8		Bread	
9		Chips	
10		Milk	
11		Yogurt	
12		Cheese	
13		Pears	
14		Celery	
15		Cliantro	
16		Potatoes	
17			
18			

Column C now systematically shows whether or not each corresponding entry in the **Grocery List** (column A) also exists within the available inventory (column B).

Analyzing the Specific Results and Data Integrity

The resultant outputs in Column C provide immediate, actionable intelligence based purely on data comparison. A result of **TRUE** confirms data overlap, while **FALSE** immediately flags a discrepancy or a missing item. Analyzing these results is straightforward and serves as a rapid quality control mechanism for any data operation.

For detailed analysis, we can review the status of several key items, understanding how the logical chain resolved the lookup:

Apples resulted in **TRUE** because the value was successfully located in the grocery inventory.
Bananas also returned **TRUE**, confirming its existence in the grocery inventory.
Carrots resulted in **FALSE**, indicating that the item was searched for using MATCH function, an error was returned, and the **NOT** function inverted the status to non-existent.
Pears returned **TRUE**, confirming the item is currently stocked.
Peppers resulted in **FALSE**, signaling its definitive absence from the inventory list.

The robust nature of this formula means that even if the dataset were massive--containing tens of thousands of rows--the integrity of the check remains intact, provided the inventory range is correctly defined as an absolute reference. Furthermore, this method is case-insensitive by default in Google Sheets, meaning that "Apples" and "apples" will typically be recognized as a match, though this behavior can be adjusted if strict, case-sensitive checking is required (usually involving the **ARRAYFORMULA** and **EXACT** functions).

Customizing Output for Enhanced Reporting with IF

While the **TRUE** and **FALSE** values are ideal for technical processing (e.g., feeding into conditional formatting or further logical functions), they may be too abstract for general reporting or end-user visualization. To enhance readability, the entire existence check formula can be seamlessly integrated into the logical test argument of the powerful IF function. This allows users to substitute the Boolean outputs with clear, descriptive text strings.

The IF function operates by evaluating a condition: if the condition (our complex lookup formula) is **TRUE**, it returns the first specified value; if **FALSE**, it returns the second specified value. Since our **NOT(ISERROR(MATCH(...)))** formula is inherently designed to return **TRUE** when an item exists, we place our "Yes" or "In Stock" message in the 'value_if_true' slot, and "No" or "Missing" in the 'value_if_false' slot.

To implement this textual customization, we modify the previous formula by enclosing it entirely within the **IF** statement, specifying "Yes" and "No" as the return values:

If you prefer to return human-readable values other than TRUE and FALSE, you can wrap the formula in an **IF** function and specify the desired return strings:

=IF(NOT(ISERROR(MATCH(A2,$B$2:$B$16,0))), "Yes", "No")

The following screenshot shows how applying this customized formula appears in practice:

C2 =IF(NOT(ISERROR(MATCH(A2,$B$2:$B$16,0))), "Yes", "No")

	A	B	C
1	**Grocery List**	**Grocery Inventory**	**Item in Grocery List Exists in Inventory?**
2	Apples	Avocado	Yes
3	Bananas	Peaches	Yes
4	Carrots	Bananas	No
5	Pears	Apples	Yes
6	Peppers	Watermelon	No
7		Flour	
8		Bread	
9		Chips	
10		Milk	
11		Yogurt	
12		Cheese	
13		Pears	
14		Celery	
15		Cliantro	
16		Potatoes	
17			
18			
19			

The customized formula now returns "Yes" if the item in the **Grocery List** exists in the **Grocery Inventory** or "No" if it does not, significantly improving the report's accessibility and clarity.

Summary of Best Practices for Existence Checks

The method of using **NOT(ISERROR(MATCH(...)))** represents the most efficient and robust technique for non-retrieval based lookups in Google Sheets. It skillfully handles the error states inherent in lookup operations, translating them into reliable logical outputs. Users are strongly advised to always double-check their absolute references (using $ symbols) on the lookup range to prevent shifting errors when propagating the formula across many rows.

By mastering this combination of functions, data professionals can automate crucial validation tasks that are otherwise tedious and error-prone when performed manually. This technique is not limited to inventory; it is equally effective for validating email lists against subscription lists, checking user IDs across systems, or ensuring compliance metrics are met across multiple data sets.

Ultimately, whether your final requirement is a simple Boolean for internal system use or a

customized textual output for stakeholder reporting, the foundational logic derived from the nested **MATCH**, ISERROR, and NOT function structure remains the optimal approach for determining if one column's values exist within another.