How to Check if All Cells Are Equal in Excel

Authored by

stats writer

November 21, 2025

The fundamental need to assess equality across numerous data points is a cornerstone of data validation and analysis, particularly within spreadsheet programs like Google Sheets and Excel. While simple comparisons between two cells are straightforward--often utilizing the basic IF function--checking for the exact same value across three, four, or dozens of cells requires more sophisticated logical constructs. This task is especially common when ensuring data consistency, verifying survey responses, or confirming uniform metrics across a sample group. Standard functions like the IF function, while useful for simple comparisons, often fall short when dealing with array-based comparisons, necessitating the combination of multiple formulas to achieve the desired Boolean data type output or customized text result.

In traditional spreadsheet logic, one might be tempted to nest multiple IF statements, or chain comparison operators (e.g., A1=B1, B1=C1, C1=D1). However, these methods quickly become cumbersome and inefficient as the number of cells increases. Fortunately, modern spreadsheet platforms provide powerful array processing capabilities that streamline this operation significantly. We will specifically focus on using the ARRAYFORMULA function alongside logical operators like AND to create a single, elegant, and scalable formula capable of evaluating simultaneous equality checks across an entire row of data. For those working in Excel, alternative approaches using the COUNTIF or COUNTIFS functions can also be employed to determine if all cells in a range match a specific criterion or a baseline cell.

The Core Formula for Array Comparison in Google Sheets

To efficiently determine if a set of cells shares the exact same value, we utilize a concentrated logical structure that leverages the power of array processing. In Google Sheets, the combination of the ARRAYFORMULA and AND functions provides a clean solution. The fundamental approach involves comparing one reference cell against an array containing all the other cells we wish to check. If the reference cell equals every element within that array, the condition for equality across the entire set is met, guaranteeing that every value is identical to the first value evaluated.

The core logical structure is built upon the premise of testing equivalency. We designate a single cell, such as **B2**, as the benchmark value. We then compare **B2** against a virtual array--created using curly braces `{}`--containing all the subsequent cells we are interested in, for example, **C2**, **D2**, and **E2**. When this comparison (`B2={C2, D2, E2}`) is executed, the spreadsheet returns a temporary array of Boolean values (e.g., {TRUE, FALSE, TRUE}), indicating the result of each individual comparison relative to **B2**. The ARRAYFORMULA is essential here, as it forces the evaluation engine to process the comparison against the entire defined array instead of just the first element, which is the default behavior for most functions.

Once the array of Boolean results is generated, the encompassing AND function takes over. The purpose of the AND function is to evaluate a set of logical arguments and return TRUE only if

every single argument in that set is also TRUE. If even one comparison within the array returns **FALSE**--meaning one cell value did not match the benchmark cell **B2**--the final output of the entire formula will be **FALSE**. This sophisticated combination guarantees that we are checking for perfect, simultaneous equality across all specified cells in a highly efficient manner, bypassing the need for lengthy chains of individual logical tests.

Implementing the Base Equality Check Formula

We can now introduce the specific formula used in Google Sheets to execute this robust comparison. If we aim to check if the values in cells **B2**, **C2**, **D2**, and **E2** are all identical, the formula is structured to explicitly compare the baseline cell (**B2**) against a literal array of the subsequent cells, ensuring that all elements are held to the same standard.

You can use the following formula in Google Sheets to check if the values in multiple cells are equal:

=AND(ARRAYFORMULA(B2={C2,D2,E2}))

This particular formula checks if the values in cells **B2**, **C2**, **D2**, and **E2** are all equal. It functions by testing whether **B2=C2**, **B2=D2**, and **B2=E2** simultaneously.

If they are all equal, the formula returns **TRUE**. Otherwise it returns **FALSE**.

This methodology is significantly cleaner than creating a lengthy formula such as `=AND(B2=C2, C2=D2, D2=E2)`, especially when dealing with a large number of columns. While the non-array formula is valid, the array method using curly braces is more scalable and conceptually aligns with advanced spreadsheet practices, making it easier to adapt to larger datasets without manually chaining dozens of logical tests. The use of ARRAYFORMULA ensures that the comparison is evaluated across all elements in the array simultaneously, yielding a concise final result.

The following example shows how to use this formula in practice.

Example: Check if Multiple Cells are Equal in Google Sheets

Consider a scenario where we are tracking the performance consistency of basketball players across four different games. Our goal is to quickly identify which players scored the exact same number of points in every game recorded. This type of consistency check requires comparing the values across four cells for each player row, ensuring data integrity across the measured period.

Suppose we have the following dataset in Google Sheets that shows the number of points scored by various basketball players during four different games:

	A	B	C	D	E
1	**Player**	**Game 1**	**Game 2**	**Game 3**	**Game 4**
2	A	10	10	10	10
3	B	12	18	18	8
4	C	14	14	13	12
5	D	14	19	13	20
6	E	15	15	15	15
7	F	13	13	13	13
8	G	19	27	27	23
9	H	25	31	30	30
10	I	24	30	40	30
11	J	20	25	23	12
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					

To perform the equality check for Player A (Row 2), which involves comparing cells B2, C2, D2, and E2, we would input the core formula into the designated evaluation column, which in this case is cell **F2**. This calculation instantly verifies the consistency of scores across the four games relative to Player A's performance in Game 1 (Cell B2). By selecting the first cell as the reference point, we establish a fixed criterion against which all subsequent cells are measured for absolute parity.

We can type the following formula into cell **F2** to check if the points values in each of the games for player A are all equal:

=AND(ARRAYFORMULA(B2={C2,D2,E2}))

A significant benefit of this formula structure is its effortless replicability across large datasets. Once entered into the first row (**F2**), we simply drag the formula handle down the remainder of column F. This action automatically adjusts the row references (from B2:E2 to B3:E3, B4:E4, and so on) for each subsequent player, allowing for rapid and accurate data validation across the entire

table without needing to manually modify the formula for every row.

We can then click and drag this formula down to each remaining cell in column F:

F2 =AND(ARRAYFORMULA(B2={C2,D2,E2}))

	A	B	C	D	E	F
1	**Player**	**Game 1**	**Game 2**	**Game 3**	**Game 4**	**All Game Values Equal?**
2	A	10	10	10	10	TRUE
3	B	12	18	18	8	FALSE
4	C	14	14	13	12	FALSE
5	D	14	19	13	20	FALSE
6	E	15	15	15	15	TRUE
7	F	13	13	13	13	TRUE
8	G	19	27	27	23	FALSE
9	H	25	31	30	30	FALSE
10	I	24	30	40	30	FALSE
11	J	20	25	23	12	FALSE
12						
13						
14						
15						
16						
17						
18						
19						

Interpreting Boolean Results (TRUE/FALSE)

The immediate output of the implemented formula is a set of Boolean data type values: **TRUE** or **FALSE**. These results are fundamental in computational logic, providing a definitive, non-ambiguous answer to the question of equality across the specified range. The result clearly communicates whether the condition--that all four game scores are identical--was met for that specific player's row.

The formula returns **TRUE** if the values in all four games are equal and **FALSE** otherwise.

Understanding this binary output is crucial for data analysis. For example, when examining the result for Player A, who scored 10 points consistently across all four games, the formula correctly evaluated the comparison as: 10=10, 10=10, 10=10. Since all components evaluated to TRUE, the final result displayed in cell F2 was **TRUE**, indicating perfect scoring consistency.

Conversely, consider the result for Player B. If Player B scored 12, 12, 15, and 12 points, the

underlying array comparison would look like: `12={12, 15, 12}`. This internal operation produces the array of Boolean values: {TRUE, FALSE, TRUE}. Because the score in Game 3 (15 points) did not match the score in Game 1 (12 points), the inclusion of a **FALSE** value within the comparison array immediately causes the encompassing AND function to return **FALSE**. This confirms the inconsistency in Player B's performance metrics, signaling that not all cells were equal.

Customizing the Output with the IF Function

While **TRUE** and **FALSE** are technically accurate and computationally efficient, users often prefer human-readable outputs, especially when sharing spreadsheets or presenting data summaries to non-technical audiences. By wrapping our sophisticated array comparison formula within the outer structure of an IF function, we can replace the standard Boolean outputs with custom text strings. This modification greatly enhances the usability and clarity of the validation column.

The IF function operates by evaluating a logical test. If the test returns **TRUE**, it returns a specified value (the value_if_true argument); if it returns **FALSE**, it returns a different specified value (the value_if_false argument). Since our entire `AND(ARRAYFORMULA(...))` structure already serves as a perfect logical test that outputs a Boolean value, we can insert it directly into the first argument of the IF function.

If you would like to return values other than **TRUE** and **FALSE**, you can wrap the formula in an **IF** function.

For example, to return the descriptive values "Equal" and "Not Equal" instead of the default Boolean results, the original formula is encapsulated. It is essential that the custom text strings are enclosed in quotation marks within the IF function syntax to denote them as literal text outputs rather than cell references or functions, thereby ensuring they are correctly displayed in the resulting cell.

For example, you can use the following formula to instead return the values **Equal** and **Not Equal**:

=IF(AND(ARRAYFORMULA(B2={C2,D2,E2})), "Equal", "Not Equal")

The following screenshot shows how to use this formula in practice:

F2 =IF(AND(ARRAYFORMULA(B2={C2,D2,E2})), "Equal", "Not Equal")

	A	B	C	D	E	F
1	**Player**	**Game 1**	**Game 2**	**Game 3**	**Game 4**	**All Game Values Equal?**
2	A	10	10	10	10	Equal
3	B	12	18	18	8	Not Equal
4	C	14	14	13	12	Not Equal
5	D	14	19	13	20	Not Equal
6	E	15	15	15	15	Equal
7	F	13	13	13	13	Equal
8	G	19	27	27	23	Not Equal
9	H	25	31	30	30	Not Equal
10	I	24	30	40	30	Not Equal
11	J	20	25	23	12	Not Equal
12						
13						
14						
15						
16						
17						
18						
19						

The formula now returns **Equal** if the values in all four games are equal and **Not Equal** otherwise.

Alternative Methods: Using COUNTIF in Excel and Sheets

While the AND(ARRAYFORMULA) technique is highly effective, especially for complex comparisons in Google Sheets, other spreadsheet programs like Microsoft Excel offer alternative strategies that rely heavily on statistical functions such as COUNTIF or COUNTIFS. These methods determine equality by checking if the count of matching values equals the total number of cells being examined, providing a different perspective on data validation.

The primary alternative method involves using the COUNTIF function. To check if a range of cells (e.g., B2:E2) all equal the value in the first cell (B2), you can count how many cells in the range match the value in B2. If that count equals the total number of cells in the range (four in this example), then all cells must be equal. The logical formula structure would look something like `=COUNTIF(B2:E2, B2) = 4`. This formula is concise and easily understandable, returning **TRUE** if all four cells match B2, and **FALSE** otherwise.

Furthermore, for scenarios where you need to check for equality across multiple criteria simultaneously or across non-contiguous ranges, the COUNTIFS function in Excel can be advantageous. Although slightly more complex to set up for a simple equality check, COUNTIFS

provides flexibility for defining multiple conditions that must all be met. The choice between the AND(ARRAYFORMULA) method and the COUNTIF approach often comes down to the user's familiarity with the platform and the specific requirements of the dataset, but both reliably perform the required multi-cell equality check with high precision.

Handling Data Types and Edge Cases

When checking for equality across multiple cells, it is critical to consider the nature of the data involved. Spreadsheet comparisons, by default, are often strict regarding data type. For instance, a cell containing the numerical value **5** might not be considered equal to a cell containing the text string **"5"**, even though they appear identical to the human eye. This distinction is crucial when building robust validation formulas, as unexpected mismatches in data type can lead to erroneous **FALSE** results.

If your dataset includes mixed data types, or if you suspect numerical values might be stored as text due to import errors or formatting issues, you may need to incorporate functions like `VALUE()` or `TEXT()` to harmonize the data types before the comparison takes place. For example, coercing all values to text before comparison ensures that the formula checks for exact character sequence equality, regardless of the underlying numeric status. Conversely, coercing text numbers into numeric values ensures mathematical equality is tested rigorously.

Furthermore, handling empty cells or zero values is another common edge case that requires careful attention. If an empty cell is included in the array comparison, the formula might treat it differently depending on the context: sometimes as a zero (in numerical contexts) or an empty string (in text contexts). This can lead to unintended **TRUE** or **FALSE** results if the other cells contain actual zeros or empty strings. To specifically exclude empty cells from the equality check, an extra layer of filtering using functions like `FILTER` or an expanded AND condition (e.g., requiring the count of cells to equal the count of non-empty cells that match the benchmark) is often necessary to ensure the calculation is mathematically sound and reflective of the intended validation logic, making the analysis more accurate.

Summary of Multi-Cell Comparison Techniques

Successfully checking for equality across multiple cells is a fundamental skill in advanced spreadsheet management and data quality control. Whether you are validating dataset integrity or ensuring uniformity in calculated metrics, the methods discussed provide powerful tools for automated checks. For Google Sheets users, the combined power of ARRAYFORMULA and AND provides the most concise and efficient method for array-based comparison, allowing for rapid deployment across large tables using simple drag-and-drop functionality.

The versatility of wrapping this core logic within the IF function allows analysts to tailor the output

for specific reporting needs, transitioning from pure logical outputs to descriptive labels like "Equal" or "Not Equal." While other spreadsheet programs may lean towards statistical counting functions like COUNTIF, the underlying principle remains the same: confirm that every evaluated cell matches a common reference point. Mastering these techniques ensures that your data validation processes are accurate, scalable, and easy to interpret, thereby increasing the reliability of your spreadsheet analysis.