# Split a Cell Vertically in Google Sheets

Authored by

**stats writer**

November 17, 2025

## Introduction to Vertical Data Transformation in Google Sheets

Effective data management often requires transforming raw input into highly structured formats suitable for analysis. A recurring scenario in spreadsheet applications is dealing with concatenated data--multiple values stored within a single cell, separated by a specific character, known as a delimiter. While the default behavior for splitting text typically results in a horizontal expansion across columns, many analytical tasks, such as creating mailing lists or preparing data for pivot tables, demand a vertical arrangement across rows.

This article provides an in-depth guide for executing a clean, valid vertical split in Google Sheets. This sophisticated maneuver leverages the combined power of two native array functions: **SPLIT** and **TRANSPOSE**. By chaining these functions, we can override the default horizontal output and convert a lengthy string into a neat, single-column list, dramatically enhancing data readability and usability.

Mastering this technique is fundamental for any advanced user of Google Sheets. It allows for efficient restructuring of large, complex data strings without relying on manual entry, scripting, or external tools, ensuring that your data preparation process is both robust and scalable. The following sections will break down the required formula and provide a detailed, practical example.

## The Core Formula for Vertical Splitting

To achieve a vertical split, we must nest the **SPLIT** function inside the **TRANSPOSE** function. The inner **SPLIT** function is responsible for parsing the text string and creating an array of values based on the specified delimiter. The outer **TRANSPOSE** function then takes this horizontally oriented array and mathematically rotates it 90 degrees, forcing the output to flow vertically down the rows.

The standard construction of this powerful combined formula requires specifying two critical pieces of information: the location of the source cell and the precise character string that separates the values. For illustrative purposes, if we assume the source data is located in cell **A2** and the items are separated by a comma followed by a space (", "), the formula is written as follows.

This particular formula is designed to split the values in cell **A2** vertically into multiple cells, using a comma and space as the delimiter to precisely determine where to break the string values:

**=TRANSPOSE(SPLIT(A2, ", "))**

## Detailed Breakdown of the SPLIT Function

The SPLIT function is the engine of this operation, designed specifically for tokenizing text. Its core purpose is to read a continuous text string and segment it into smaller pieces based on the

presence of a specified delimiter. The syntax for **SPLIT** is generally `SPLIT(text, delimiter)`, where `text` is the cell reference containing the source data, and `delimiter` is the character or string used to mark the separation points.

When the SPLIT function executes independently, it always returns its result as a horizontal array, meaning the separated parts are placed into adjacent columns starting from the cell where the formula is entered. This output behavior is inherent to how spreadsheet functions handle text parsing arrays. If the string contains four items, the result will occupy one row and four columns.

It is vital to specify the delimiter accurately. If the delimiter includes a space, such as `", "`, that precise combination must be used in the formula. If only `","` is used when a space follows the comma, the resulting text segments will retain the unwanted leading space, leading to potential issues in subsequent data processing steps (like filtering or matching). The **SPLIT** function effectively generates the necessary data components; it simply needs **TRANSPOSE** to organize them correctly for vertical display.

## The Role of the TRANSPOSE Function in Array Manipulation

The TRANSPOSE function is a matrix manipulation tool whose utility extends far beyond simple data rearrangement. Its primary function is to interchange the rows and columns of an array or range of cells. This capability is exactly what is needed to convert the horizontal output of **SPLIT** into a vertical structure.

When the 1xN horizontal array generated by the **SPLIT** function is passed to TRANSPOSE, the function treats the single row as N separate columns and transforms them into N separate rows while maintaining one column. This conversion instantly shifts the data orientation from horizontal to vertical, thereby fulfilling the requirement of splitting a cell's contents across multiple rows.

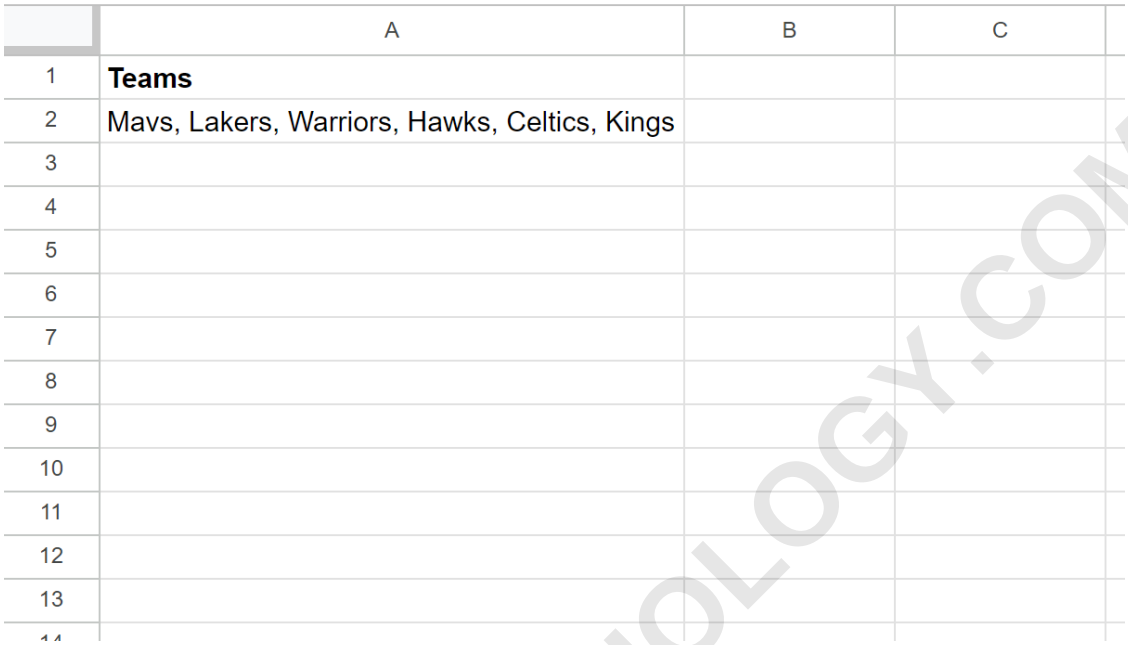
Because **TRANSPOSE** is applied directly to the array output of **SPLIT**, the process is instantaneous and dynamic. Any change to the source cell (**A2** in our example) automatically updates the split vertical list, maintaining data consistency without requiring manual intervention. This dynamic linking underscores why using array formulae is the preferred methodology for complex data restructuring in Google Sheets.

## Practical Example: Demonstrating Vertical Split

To solidify the concept, let us work through a concrete example. Suppose we are tracking sports teams, and the team names have been inadvertently compiled into a single cell, **A2**. The names are separated by a comma followed by a space. This consolidated list prevents us from individually analyzing or counting the teams.

We want to separate these names so that each team occupies its own row, starting in column **C**, thereby making the data suitable for creating a proper list or database entry.

The initial state of the data shows all team names contained within cell **A2**, as illustrated below:



| | A | B | C |
|---|---|---|---|
| 1 | **Teams** | | |
| 2 | Mavs, Lakers, Warriors, Hawks, Celtics, Kings | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |



Our objective is to apply the combined **TRANSPOSE(SPLIT())** function to transform this single cell into a clean, vertically stacked list, which is the foundational step for virtually all subsequent data analysis.

## Step-by-Step Implementation and Execution

The implementation begins by selecting the starting cell for the output, which, in this case, is cell **C2**. This single cell will house the complete array formula responsible for generating the entire vertical list.

**Determine the Parameters:** We identify the source cell as **A2** and confirm the required delimiter is `", "`.

**Input the Nested Formula:** We type the following combined formula directly into cell **C2**, ensuring that all parentheses are properly matched and the delimiter is correctly quoted:

**=TRANSPOSE(SPLIT(A2, ", "))**

Upon execution, Google Sheets calculates the **SPLIT** result (the temporary horizontal array) and immediately uses the TRANSPOSE function to pivot this array, causing the individual team names

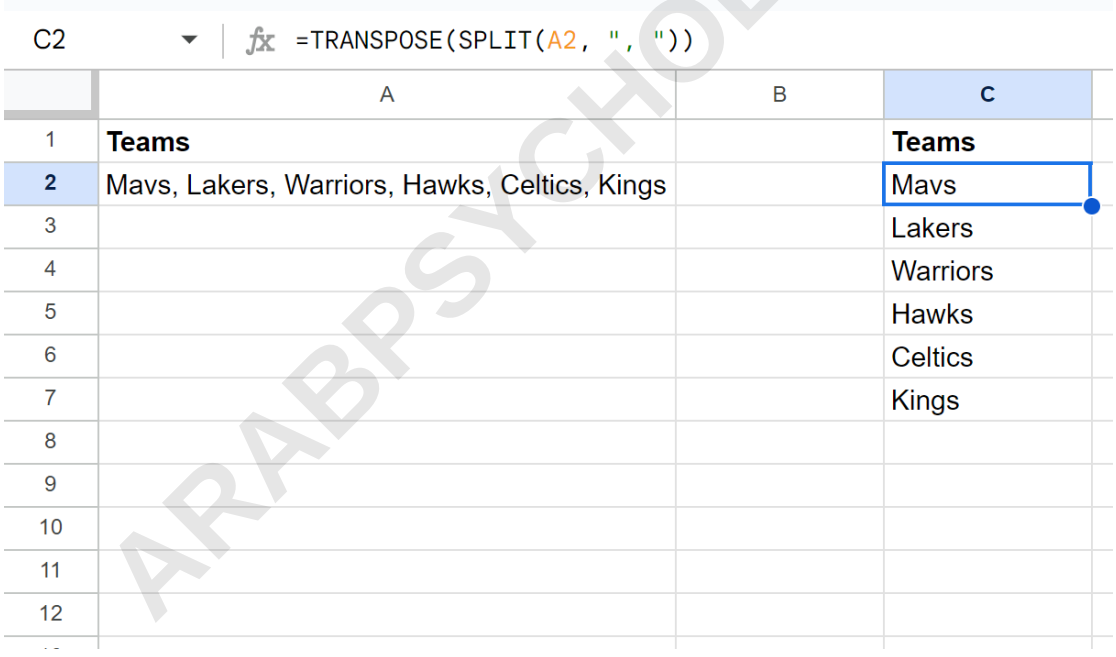
to populate cells **C2**, **C3**, **C4**, and so on. The use of a single cell for the formula entry simplifies management and reduces potential spreadsheet errors.

## Reviewing the Vertical Array Output

The resulting data structure in column C confirms the successful application of the vertical split. The values previously cramped into cell **A2** are now dynamically displayed in separate rows. It is important to recognize that the array output occupies these subsequent cells, meaning only cell **C2** actually contains the driving formula.

If you select cells below **C2**, such as **C3** or **C4**, you will observe that the input bar does not display editable content but rather indicates that the cell is occupied by the array result originating from **C2**. This is the expected behavior of array formulae, where a single formula generates results that spill across multiple cells.

The following screenshot illustrates the final, clean output where the concatenated text has been reorganized into a structured vertical list:




This vertically separated data is now primed for filtering, sorting alphabetically, or integration into pivot table reports, tasks that were impossible when the data was confined to cell **A2**.

## Customizing for Different Delimiters

While our example used `", "`, the versatility of the **SPLIT** function allows it to handle nearly any separation character or string. The user must be vigilant in identifying the exact delimiter used in

their source data and ensuring it is accurately represented as the second argument in the **SPLIT** function.

Consider alternative delimiters and their corresponding formula changes:

If values are separated by a semicolon only: `=TRANSPOSE(SPLIT(A2, ";"))`

If values are separated by a pipe character: `=TRANSPOSE(SPLIT(A2, "|"))`

If values are separated by a unique multi-character string (e.g., `::`): `=TRANSPOSE(SPLIT(A2, "::"))`

It is crucial to enclose the delimiter in double quotation marks, regardless of whether it is a single character or a complex string. If the source data is messy or inconsistent, cleaning the source cell first, perhaps using the **SUBSTITUTE** function to replace multiple inconsistent separators with a single, uniform one, can prevent errors in the **SPLIT** process.

## Handling Potential Errors and Advanced Usage

When implementing this array formula, the most common issue encountered is the `#REF!` error. This error occurs specifically when the dynamic array generated by the formula attempts to write its output into cells that are already occupied by other data. Since the array needs a contiguous empty range below the starting cell, ensure that all intended output rows are clear before entering the formula.

For users dealing with entire columns of concatenated data (e.g., A2, A3, A4, etc., all need splitting), this technique can be enhanced by wrapping the entire structure in an **ARRAYFORMULA**. This allows the single formula to dynamically apply the **TRANSPOSE(SPLIT())** logic to every row in the specified range simultaneously, streamlining the data cleanup process across hundreds or thousands of rows.

The combination of TRANSPOSE and SPLIT is a cornerstone technique for efficient data manipulation in Google Sheets. By understanding how to pivot array outputs, spreadsheet users gain robust control over data flow and structure.

**Note**: Comprehensive documentation for the **SPLIT** function, including optional parameters such as those handling empty cells or splitting by each character, can be found in the official SPLIT function documentation provided by Google.