How to Split Comma-Separated Values into Rows in Excel: A Step-by-Step Guide

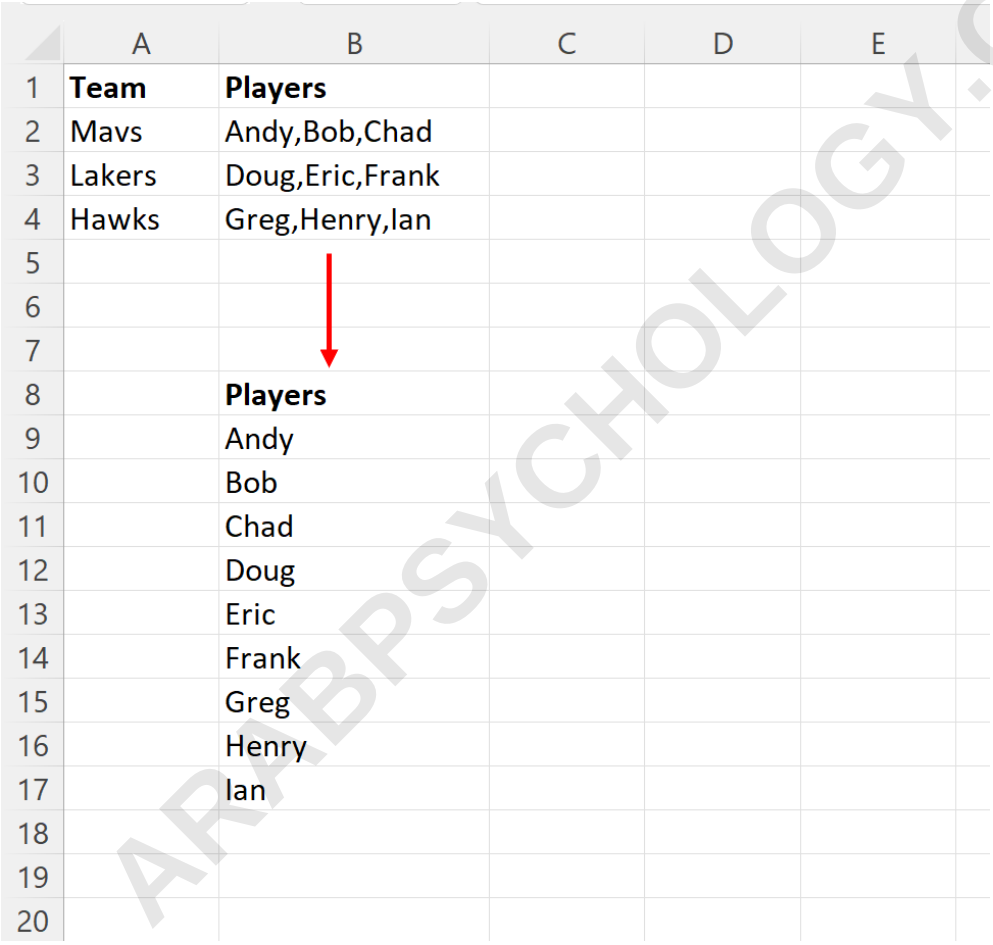
Authored by

stats writer

January 18, 2026

In data analysis, it is extremely common to encounter datasets where multiple entries are consolidated into a single cell, typically separated by a delimiter like a comma. These comma-separated values (CSV) pose a challenge when you need to process each item individually for proper analysis or reporting. Fortunately, modern versions of Microsoft Excel provide powerful dynamic array functions that make splitting these values into distinct rows seamless, transforming a wide dataset into a long, analytical format.

The goal of this operation is to normalize the dataset, ensuring that every individual data point (in this case, a player name) occupies its own unique row, along with its corresponding category (the team name). Consider the typical outcome we aim to achieve:



This comprehensive, step-by-step guide will walk you through the necessary functions--TEXTSPLIT, TRANSPOSE, and VSTACK--to achieve this complex data restructuring efficiently, minimizing manual effort and potential errors.

Introduction: Understanding the Challenge of CSV Data in Excel

Data normalization requires that there is no repetition of grouping variables and that all atomic data

elements are stored in separate fields. When multiple values, such as a list of product codes or employee skills, are crammed into a single cell using a comma delimiter, standard Excel features like PivotTables or VLOOKUP become ineffective, as they cannot easily parse the grouped data.

The traditional method for handling this involved using the "Text to Columns" wizard repeatedly, which is a destructive process that overwrites source data and is highly impractical for large datasets. Furthermore, it only splits data horizontally, still requiring extensive manual cleanup and vertical concatenation.

Modern Excel, particularly versions supporting dynamic arrays (Excel 365 and Excel 2021), simplifies this dramatically. By chaining together specialized functions, we can create a fully dynamic solution that automatically updates whenever the source data changes, providing a robust and flexible workflow for handling delimited data sets.

Prerequisites and Dataset Preparation

To successfully follow this tutorial, you must be using a version of Microsoft Excel that supports the dynamic array architecture, specifically including the TEXTSPLIT and VSTACK functions. These functions eliminate the need for cumbersome intermediate steps that were previously unavoidable.

The chosen example simulates a common business scenario where master records (Team Name) are associated with sub-records (Player Names) listed in a single, delimited field. The principle demonstrated here can be applied universally to inventory tracking, project management tasks, or any other data where lists are stored non-atomically.

Ensure your working environment is set up with sufficient adjacent columns available, as the initial splitting step will spill the data horizontally before we consolidate it vertically. We will use columns A and B for the input and columns D, E, F, and subsequent columns temporarily for calculation, before outputting the final result below the source data.

Step 1: Entering and Reviewing the Source Data

Before diving into the complex formulas, the first crucial step involves properly structuring and entering the source data. We are working with a scenario where different teams have listed their players in a single cell, separated by commas. This initial setup is typical of data exported from basic databases or forms where field separation was not enforced during entry.

For this tutorial, we will use a small dataset representing basketball teams and their respective player rosters. Enter this data into your Excel sheet, starting at cell **A1**, ensuring the player names in Column B are correctly separated by the chosen delimiter, which in this case is the comma (`,`). Note that consistent spacing after the comma is important if you want the output to be clean,

though advanced users can incorporate functions like TRIM within TEXTSPLIT to automatically remove leading or trailing spaces.

The dataset should resemble the following structure, demonstrating the challenge we intend to solve:

	A	B	C	D	E
1	**Team**	**Players**			
2	Mavs	Andy,Bob,Chad			
3	Lakers	Doug,Eric,Frank			
4	Hawks	Greg,Henry,Ian			
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

Our goal is to transform the contents of Column B so that each individual player name occupies its own unique row, retaining the associated Team Name from Column A for every new entry created.

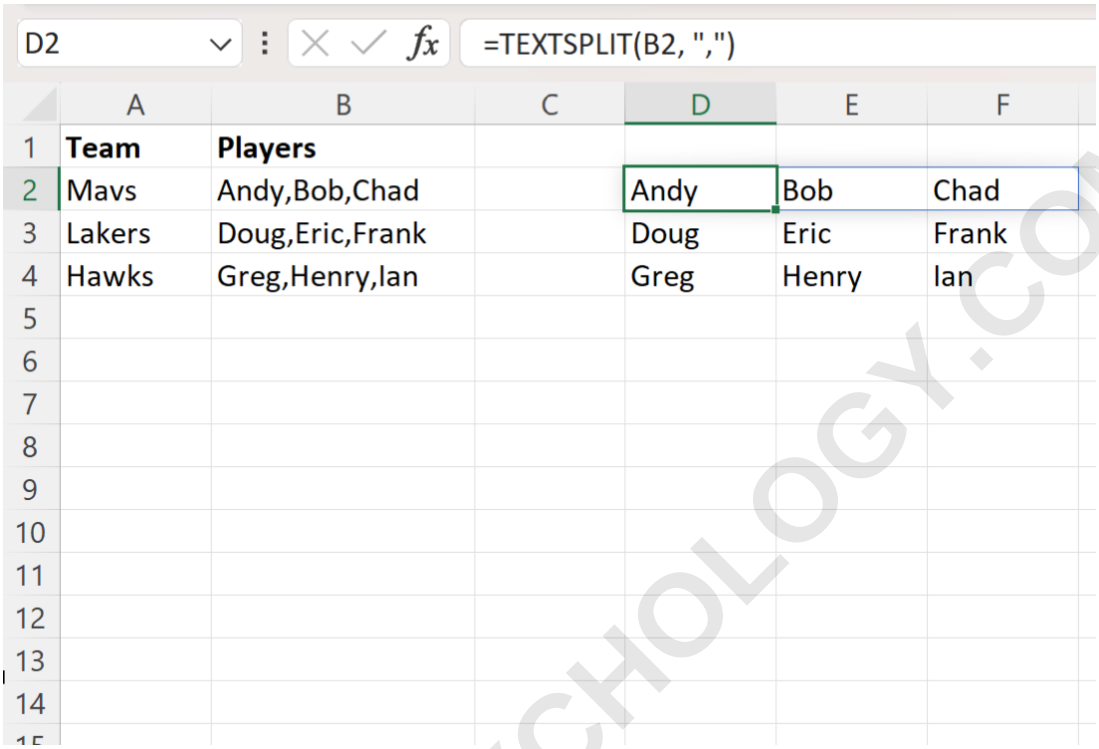
Step 2: Utilizing the TEXTSPLIT Function for Column Separation

The core mechanism for breaking apart the CSV string is the powerful TEXTSPLIT function. Introduced in recent versions of Excel, TEXTSPLIT handles the common task of parsing strings based on a specific delimiter and automatically spills the results across adjacent columns, functioning as a dynamic array.

We begin by implementing this formula in cell **D2**. This calculation will analyze the comma-separated list found in cell **B2** and distribute the individual player names horizontally across Column D, E, F, and so forth. Enter the following precise formula into **D2**:

=TEXTSPLIT(B2, ",")

After entering the formula in **D2**, you will observe the results immediately spreading out. Since the TEXTSPLIT function is a dynamic array function, Excel automatically manages the required output range. Next, copy this formula down to the rest of the relevant rows (in our case, **D3** and **D4**) to apply the splitting operation to all teams in the dataset.



At this stage, the data is successfully separated but remains in a "wide" format--each team's players are spread across columns D, E, and F. The subsequent steps are dedicated to converting this horizontal layout into the desired vertical structure for proper normalization.

Detailed Look at TEXTSPLIT Syntax and Parameters

Understanding the syntax of the TEXTSPLIT function is vital for effective data parsing, especially when dealing with complex delimiters or cleaning requirements. The basic required arguments are the text to be split (e.g., `B2`) and the column delimiter (e.g., `","`).

The full syntax is `TEXTSPLIT(text, col_delimiter, , , , )`. While we only used the first two arguments, advanced applications might require the optional parameters:

: This allows splitting based on both column and row delimiters simultaneously, generating a true two-dimensional array directly from a single string.

: Setting this to `TRUE` (1) instructs Excel to skip empty strings resulting from multiple consecutive delimiters (e.g., if a cell contains `"Item A,,Item C"`). Using this is highly recommended for

maintaining data cleanliness.

: Allows you to specify a value (like `"NA"` or `0`) to fill in missing cells if the resulting arrays are jagged (i.e., they have differing numbers of elements).

For our specific scenario, simply defining the comma as the column delimiter is sufficient to isolate each player name into its own temporary holding cell, preparing the data for the transposition stage.

Step 3: Employing TRANSPOSE to Reorient the Data Matrix

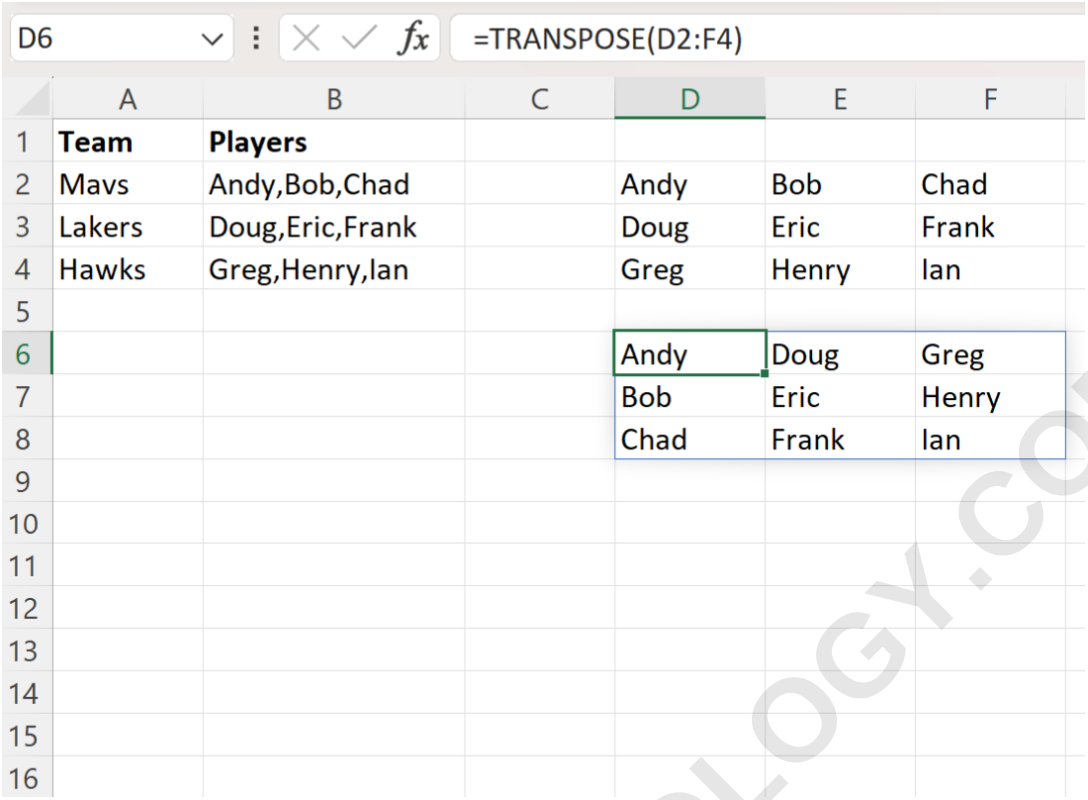
We have successfully separated the names horizontally, but our ultimate goal is a vertical list where all players stack one below the other. This requires rotating the temporary data matrix we created in the previous step (range D2:F4). The TRANSPOSE function is perfectly suited for this task, as it flips a range of cells from columns to rows, or vice versa, by exchanging row and column indices.

We will apply the TRANSPOSE function starting in cell **D6**. The argument for this function is the entire range containing the split player names, which is **D2:F4**. Enter the following formula into **D6**:

=TRANSPOSE(D2:F4)

Because TRANSPOSE is also a dynamic array function, the resulting 3x3 matrix will automatically spill across the necessary rows and columns starting from D6. The three rows (Team 1, Team 2, Team 3) are now transformed into three columns (D6:D8, E6:E8, F6:F8), where each column represents the complete list of players from one original team.

The result of this transposition step is crucial. While the data is still not in its final single-column format, it is now arranged in vertical stacks that directly correspond to the order needed for vertical merging. Observe the transformation in the screenshot below:



Step 4: Stacking the Results Vertically using VSTACK

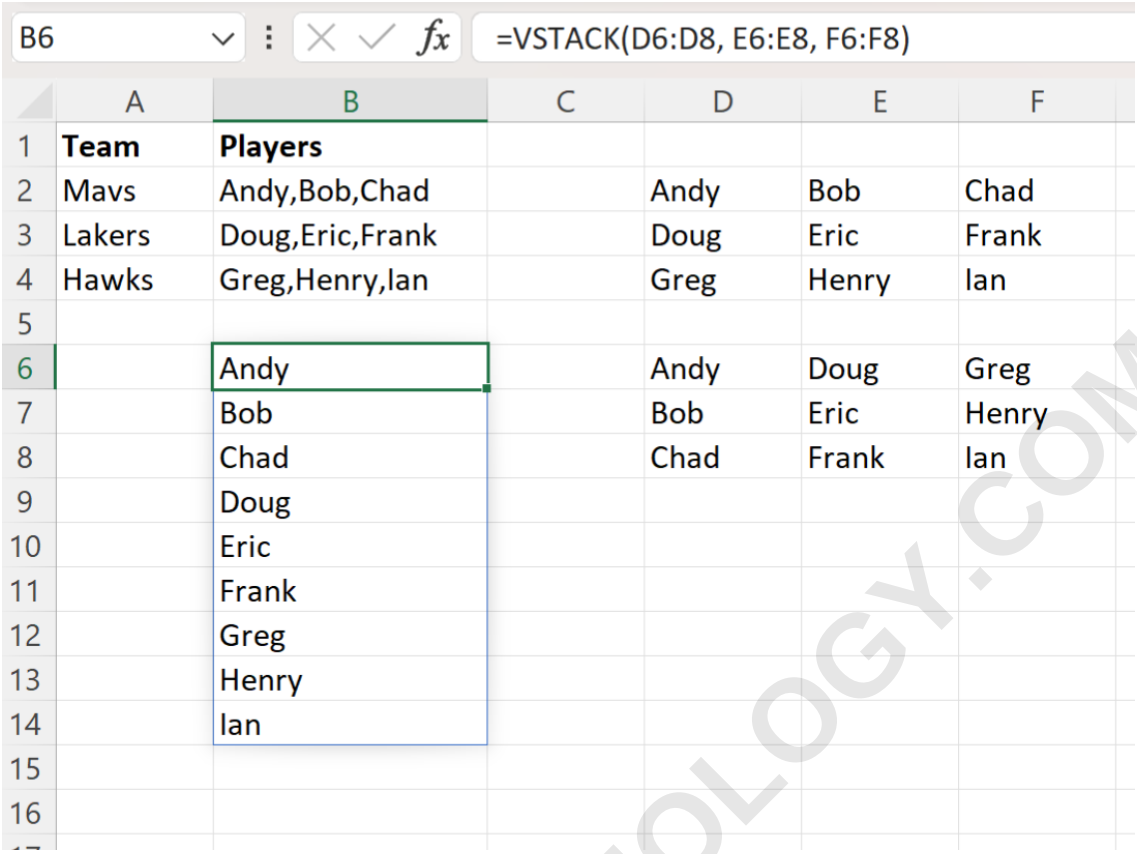
The final, and most satisfying, step involves consolidating all the vertically oriented player lists into one continuous column. For this aggregation, we employ the VSTACK function. VSTACK is designed specifically to append arrays vertically, stacking them one on top of the other in the order specified by the arguments.

We will place the resulting output in cell **B6**, ensuring it does not overwrite our original data in rows 1-4. The formula takes each column generated by the TRANSPOSE step (D6:D8, E6:E8, F6:F8) and stacks them sequentially. Enter the following formula into **B6**:

=VSTACK(D6:D8, E6:E8, F6:F8)

This single formula instantly generates the complete list of all player names, seamlessly flowing into a single vertical column. The output in Column B, starting from B6, now contains all nine player names derived from the original three rows of CSV data.

Reviewing the result confirms the successful application of the dynamic functions:



	A	B	C	D	E	F
1	**Team**	**Players**				
2	Mavs	Andy,Bob,Chad		Andy	Bob	Chad
3	Lakers	Doug,Eric,Frank		Doug	Eric	Frank
4	Hawks	Greg,Henry,Ian		Greg	Henry	Ian
5						
6		Andy		Andy	Doug	Greg
7		Bob		Bob	Eric	Henry
8		Chad		Chad	Frank	Ian
9		Doug				
10		Eric				
11		Frank				
12		Greg				
13		Henry				
14		Ian				
15						
16						

We have now successfully split the comma-separated player names from the original dataset into their own rows, achieving the normalized data structure required for advanced sorting and analysis.

Associating Metadata: Replicating Team Names

While Step 4 successfully created a clean list of players in a single column (starting at B6), a critical step for complete data analysis is ensuring that each player record is correctly linked back to its originating metadata--the team name. When splitting a single row into multiple output rows, the associated data must be replicated the appropriate number of times.

Since each of our original teams had three players, we need to create a repeating sequence: Team A three times, Team B three times, and Team C three times. This requires creating a corresponding array that matches the dimensions of our final VSTACK output (B6:B14, which is 9 rows tall).

The most straightforward method in this fixed-size example is to manually structure the repetition using VSTACK on the team names themselves, referencing the team name cell three times for each team:

=VSTACK(A2:A2, A2:A2, A2:A2, A3:A3, A3:A3, A3:A3, A4:A4, A4:A4, A4:A4)

If placed in cell **A6**, this formula replicates the team names perfectly alongside the corresponding players list in Column B. For scenarios where the number of separated values per row is variable, advanced users must use dynamic counting functions (like COUNTA combined with TEXTSPLIT and SEQUENCE) to automate the exact number of times the team name needs to be repeated before the next team name starts. This ensures the output remains dynamically linked even if player rosters fluctuate.

Combining Functions for Advanced Scenarios and Cleanup

While the multi-step process--TEXTSPLIT, TRANSPOSE, and VSTACK--clearly isolates the functionality of each dynamic array function, expert Excel users often consolidate these steps into a single, comprehensive formula. Combining these functions avoids the need for temporary helper columns (D, E, F, etc.), resulting in a cleaner and more memory-efficient workbook.

A single, consolidated approach for splitting and stacking would look like this, performing the split and immediate transposition for each row (B2, B3, B4) and then using VSTACK to aggregate them:

=VSTACK(TRANSPOSE(TEXTSPLIT(B2,",")), TRANSPOSE(TEXTSPLIT(B3,",")), TRANSPOSE(TEXTSPLIT(B4,",")))

Using the multi-step approach demonstrated in this guide, however, offers significantly better transparency and easier debugging, especially when first learning these powerful dynamic array functions. The key advantage of moving away from older methods like "Text to Columns" is the creation of a non-destructive and highly automated solution.

Conclusion and Further Data Transformation Techniques

Restructuring data from a wide, delimited format into a long, row-based format is a fundamental requirement in data preparation for analytical tools. By mastering the sequence of TEXTSPLIT, TRANSPOSE, and VSTACK, users can automate a process that was once tedious and error-prone. These dynamic array functions represent a significant leap forward in Excel's capabilities, providing powerful tools for data normalization.

It is important to note that the functionality demonstrated relies on having access to Excel versions that support these modern dynamic arrays (Excel 365 or Excel 2021). Users on older versions may need to resort to the older combination of "Text to Columns," copy/paste special (Transpose), and standard concatenation or VBA scripting, which lack the dynamic responsiveness of this formula-based approach.

For detailed technical specifications regarding the functions used, particularly the powerful vertical stacking capabilities, we encourage reviewing the official documentation.

Note: You can find the complete documentation for the **VSTACK** function in Excel here.

The following tutorials explain how to perform other common operations in Excel:

How to use the HSTACK function for horizontal data concatenation when merging arrays side-by-side.

Techniques for ensuring data quality by cleaning white space and non-printing characters using the **TRIM** and **CLEAN** functions.

Mastering Power Query for data transformation when dealing with significantly larger datasets or data sourced externally.

By mastering these advanced formula-based techniques, you ensure your data is always in the optimal format for sophisticated analysis, filtering, and reporting.