

# How to Split Comma-Separated Values into Rows in Excel

Authored by  
**stats writer**

February 2, 2026

## RECOMMENDED CITATION

stats writer (2026). *How to Split Comma-Separated Values into Rows in Excel*.  
PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=129159>

This comprehensive tutorial outlines the advanced techniques required for effectively splitting comma-separated values (CSV) contained within a single cell into individual, corresponding rows in Excel. This method utilizes modern dynamic array functions, providing a robust and efficient solution for data cleaning and preparation. Data sets often arrive in formats where related entities are condensed into one cell—a common scenario that hinders effective data analysis and reporting. By separating each value onto its own designated row, users achieve a standardized, normalized data structure that is far easier to filter, sort, and analyze.

This guide provides clear, step-by-step instructions and practical examples using powerful functions like TEXTSPLIT, TRANSPOSE, and VSTACK. Mastering this technique is crucial for data professionals and analysts working frequently with imported or uncleaned data, enabling significant improvements in data management workflows.

We aim to demonstrate how these dynamic array formulas seamlessly work together to transform complex, aggregated data into a clean, actionable format, making data manipulation significantly more straightforward.

## Excel: Split Comma-Separated Values into Rows

Working with data often presents challenges, particularly when multiple related items are concatenated into a single cell, delimited by characters like commas, semicolons, or pipes. This is a common situation when importing raw data, especially when dealing with comma-separated values (CSV) files or database exports where normalization hasn't occurred. To perform proper data analysis, filtering, or pivot table creation, we must ensure that each distinct item occupies its own row, paired with its corresponding identifier (the team name, in our example).

Fortunately, recent updates to Excel have introduced dynamic array functions that simplify this complex data transformation task dramatically. Unlike older methods requiring the 'Text to Columns' feature followed by manual restructuring, this approach is entirely formula-based, making it dynamic, scalable, and fully reactive to changes in the source data. The following steps detail a powerful combination of TEXTSPLIT, TRANSPOSE, and VSTACK that achieves the desired row separation efficiently. This transformation process is essential for anyone dealing with complex relational data structure within a spreadsheet environment.

Consider the typical scenario illustrated below, where player names for multiple teams are aggregated into individual cells in Column B. Our goal is to expand this structure so that 'Team A' is repeated for each player listed under it, with each player name occupying a unique row:

	A	B	C	D	E
1	<b>Team</b>	<b>Players</b>			
2	Mavs	Andy,Bob,Chad			
3	Lakers	Doug,Eric,Frank			
4	Hawks	Greg,Henry,Ian			
5					
6					
7					
8		<b>Players</b>			
9		Andy			
10		Bob			
11		Chad			
12		Doug			
13		Eric			
14		Frank			
15		Greg			
16		Henry			
17		Ian			
18					
19					
20					

We will proceed through a series of logical steps, leveraging the power of modern array formulas available in Microsoft 365 versions of Excel.

### Step 1: Preparing the Source Data

The foundation of any successful data manipulation task lies in having a clearly structured input dataset. For this tutorial, we are using a simple dataset featuring team identifiers and corresponding lists of player names. This scenario is highly representative of real-world data cleanup tasks where imported data needs transformation before analysis.

To begin, please ensure your spreadsheet contains the following structure. We will assume the team names are in Column A and the comma-separated values (CSVs) are in Column B, starting from row 2. This initial setup is crucial as all subsequent formulas will reference these specific cells.

**Column A (Team):** Contains the unique identifier for the group (e.g., Team A, Team B, Team C).

**Column B (Players CSV):** Contains the aggregated player names, separated by commas.

The dataset should look like this:

	A	B	C	D	E
1	<b>Team</b>	<b>Players</b>			
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					

Understanding the structure of the input data is vital. Notice that the number of players varies per team, making a static copy-paste approach impractical. We need a dynamic solution capable of handling varying lengths of delimited lists, which is precisely what the dynamic array functions provide.

## Step 2: Leveraging the TEXTSPLIT Function for Horizontal Expansion

The first major hurdle is separating the individual player names from the combined string within Column B. The traditional method involved using the 'Text to Columns' wizard, which is static and requires manual repetition. Modern Excel simplifies this immensely using the **TEXTSPLIT** function, a dynamic array function designed explicitly for dividing text strings based on a specified delimiter.

The **TEXTSPLIT** function takes the text to be split, the column delimiter, and optionally, a row delimiter. Since we want the resulting values to spread horizontally across new columns, we only specify the column delimiter (the comma, in this case). This function will spill the results into adjacent cells, adjusting automatically based on the number of items found.

Type the following formula into cell **D2**. This is the first step in converting the single cell string into a horizontal array:

**=TEXTSPLIT(B2, ",")**

Let's break down the formula parameters:

**B2:** This is the text string containing the player names we want to split.

**",":** This is the column delimiter. Since our values are comma-separated, we use the comma enclosed in double quotes.

After entering the formula in D2, you will observe the player names for the first team immediately spilling into cells D2, E2, and F2. You must then click and drag this formula down to apply it to each remaining row in column D (D3 and D4). This step generates a rectangular array of data where each row corresponds to an original team, and the columns contain the individual split values.

	A	B	C	D	E	F
1	<b>Team</b>	<b>Players</b>				
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						
7						
8						
9						
10						
11						
12						
13						
14						
15						

At this stage, the data is split horizontally, but it still maintains the original structure (one row per team). The next critical step involves reorganizing this expanded data set so that all individual names are stacked into a single, contiguous column, preparing the data for final aggregation.

### Step 3: Restructuring Data with TRANSPOSE

Once the data is split horizontally across multiple columns (D2:F4), we need a method to reorganize it into a single column structure. If we were to simply concatenate the rows, we would still lose the necessary association between the player and their original team. The most efficient way to achieve this restructuring while preserving the spatial relationship for the next step is to use

the **TRANSPOSE** function.

The **TRANSPOSE** function flips the orientation of a given array or range. Rows become columns, and columns become rows. By transposing the entire resulting array (D2:F4), we convert the structure from three rows by three columns into three columns by three rows. This intermediary step might seem counter-intuitive, but it sets the stage perfectly for the final vertical stacking operation, ensuring that the players from Team A, Team B, and Team C are grouped together vertically before being combined.

Type the following formula into a designated empty cell, such as **D6**, to transpose the data resulting from the **TEXTSPLIT** step. The range D2:F4 represents the entire output of the split names:

**=TRANSPOSE(D2:F4)**

The resulting output array, starting in D6, will now have the players listed vertically within their respective team columns (D6:D8 for Team A players, E6:E8 for Team B players, and F6:F8 for Team C players). Note that where the original data had fewer than three players (Team B and C), the **TEXTSPLIT** function might introduce empty or error values, which will also be transposed. These can often be handled later with filtering, or if necessary, refined within the initial **TEXTSPLIT** formula using the `ignore_empty` argument.

	A	B	C	D	E	F
1	<b>Team</b>	<b>Players</b>				
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	Doug	Greg
7				Bob	Eric	Henry
8				Chad	Frank	Ian
9						
10						
11						
12						
13						
14						
15						
16						

Observe the layout carefully: Team A's players occupy column D, Team B's players occupy column E, and so on. This structure is ideal for the final stacking operation, as it groups all related records vertically.

#### Step 4: Consolidating Data into Rows using VSTACK

The final and most crucial step in achieving the desired row-per-value format is consolidating the vertically arranged columns into a single, cohesive column. This is accomplished using the **VSTACK** function, another powerful member of the Excel dynamic array family. The **VSTACK** function stands for Vertical Stack and allows users to append arrays or ranges one on top of the other, effectively merging them into a single column.

By using **VSTACK** on the transposed columns (D6:D8, E6:E8, F6:F8), we combine all the player lists sequentially: first Team A's players, then Team B's, and finally Team C's. This results in one long column containing every individual player name extracted from the original CSV strings.

Type the following formula into cell **B6** (or any desired output starting cell) to execute the vertical stacking:

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

The range references (D6:D8, E6:E8, F6:F8) correspond exactly to the output ranges generated by the **TRANSPOSE** function in the previous step. The **VSTACK** function collects these arrays and outputs them into a vertical list, achieving the goal of splitting the comma-separated data into individual rows.

	A	B	C	D	E	F
1	<b>Team</b>	<b>Players</b>				
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						

Upon successful execution, Column B (starting at B6) will contain the finalized list of individual players. We have now successfully transformed the aggregated player names from the original dataset into a normalized structure where each player occupies their own dedicated row.

**Note:** You can find the complete documentation for the **VSTACK** function in [Excel's](#) official documentation.

### Step 5: Addressing Associated Data (Handling Team Names)

While the previous steps successfully extracted and stacked the player names, the resulting list (starting in B6) lacks the critical context: the team associated with each player. In a truly normalized dataset, every extracted value must be paired with its original identifier (the team name from Column A).

To replicate the team names, we must calculate how many times each team name needs to be repeated. Since the number of repetitions corresponds to the number of players found for that team, this calculation needs to be derived dynamically from the results of the [TEXTSPLIT](#) step.

A more advanced, consolidated formula would incorporate a function like **ROWS** or **COUNTA** combined with **HSTACK** and **REPT** to achieve the final two-column output (Team | Player) in a

single dynamic array formula. However, following our current step-by-step approach, you would manually recreate the team list in an adjacent column (e.g., Column A starting at A6) by counting the number of non-blank entries for each team in the transposed array (D6:F8) and repeating the corresponding original team name (A2, A3, A4) that many times. For instance, if Team A has three players (D6:D8), 'Team A' must be listed three times in A6:A8.

## Step 6: Optimizing the Workflow with a Single Formula (Advanced)

The multi-step approach using intermediate columns (D, E, F) is excellent for understanding the logic, but for experienced Excel users, the entire process can be condensed into a single, powerful dynamic array formula, eliminating the need for helper columns.

The goal is to generate two arrays simultaneously: the list of repeated team names, and the list of players, and then join them horizontally using the **HSTACK** function. This often involves functions like **BYROW**, **MAP**, or **REDUCE**, combined with **FILTER** to remove blanks.

A simplified example of the conceptual flow for a single formula solution would look like this:

Use **TEXTSPLIT** on the entire range (B2:B4) to split all values at once, resulting in a complex array.

Use **TOCOL** (or a combination of **TRANSPOSE** and **VSTACK**) to convert this array into a single column, filtering out any empty cells that resulted from uneven list lengths.

Generate the corresponding team names using functions like **MAP** and **REPT** based on the count of elements generated in step 2.

Combine the two results using **HSTACK**.

While highly efficient, these consolidated formulas are significantly more complex to debug. The three-step process outlined (**TEXTSPLIT**, **TRANSPOSE**, **VSTACK**) remains the clearest pedagogical approach for understanding the underlying data manipulation logic required in Excel.

## Step 7: Conclusion and Data Management Implications

The ability to dynamically split comma-separated values into separate rows represents a major advancement in data preparation within Excel. By utilizing the modern dynamic array functions—**TEXTSPLIT**, **TRANSPOSE**, and **VSTACK**—users can move beyond static data cleaning methods and implement solutions that automatically update as source data changes. This efficiency gain is particularly valuable in fields requiring frequent importation and normalization of large datasets for statistical processing or data analysis.

Key takeaways from this methodology include:

**Dynamic Nature:** The resulting output table updates automatically whenever the input data in the

original CSV cells is modified, ensuring data integrity without manual intervention.

**Scalability:** This method handles varying list lengths efficiently, unlike older, manual workarounds that often fail when the number of items in a cell changes.

**Clarity of Logic:** Breaking the process into three distinct steps (split horizontally, transpose for vertical grouping, stack vertically) clearly maps the necessary data transformation journey.

Mastering these functions is essential for maximizing productivity in modern spreadsheet environments. The resulting normalized data structure (one piece of information per row) is the standard prerequisite for creating powerful Pivot Tables, running advanced filtering, and generating accurate reports.

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

ARABPSYCHOLOGY.COM