How to Convert Comma-Separated Values to Rows in Excel

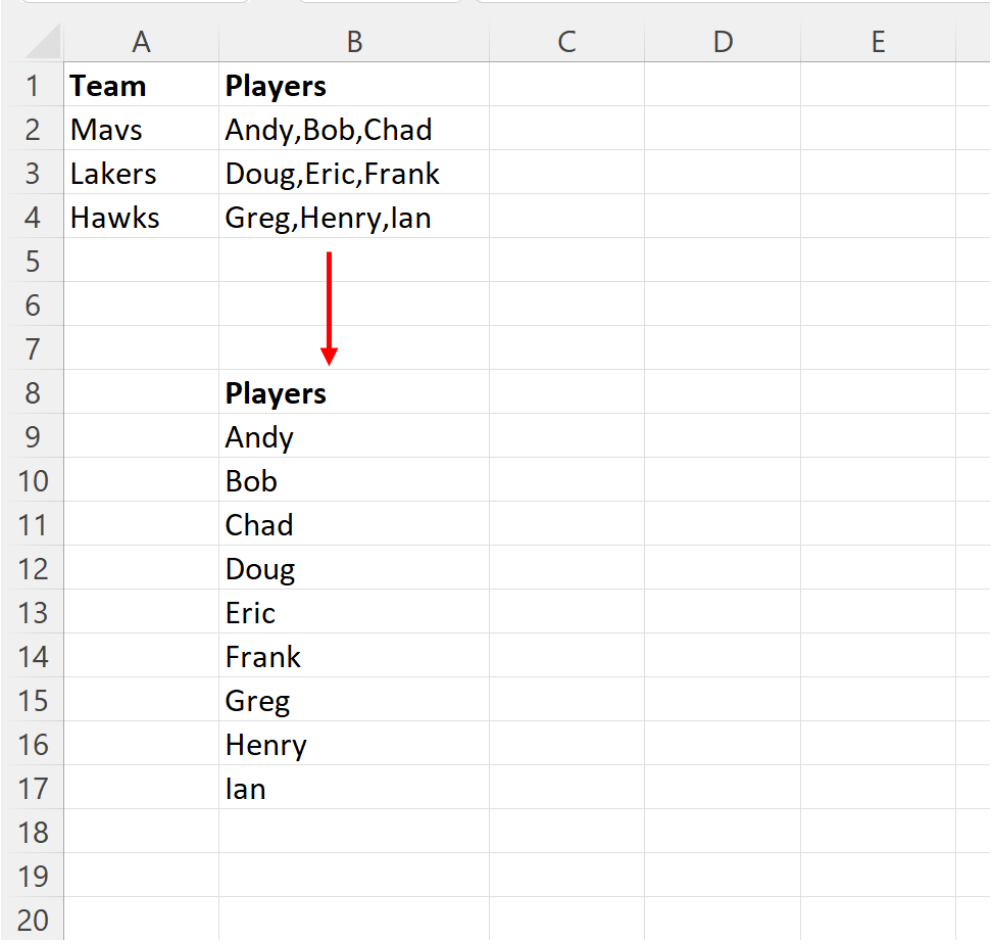
Authored by

Mohammed looti

November 11, 2025

In the realm of modern data management and analysis, users of Microsoft Excel frequently encounter a critical challenge: transforming horizontally organized, aggregated data into a vertical, row-based structure. This issue commonly arises when importing or pasting external datasets where single cells contain multiple values delimited by a comma--a format universally known as comma-separated values (CSV). For any meaningful analytical operations, such as generating accurate pivot tables, implementing effective sorting, or conducting detailed statistical analysis, the necessity to separate these aggregated values into individual, distinct rows is paramount.

This article serves as an expert, comprehensive, step-by-step tutorial demonstrating the most efficient method to achieve this transformation. We will harness the power of modern Dynamic array formulas available in recent versions of Excel (Microsoft 365 or Excel 2021 and later). This approach entirely bypasses older, often convoluted processes that relied on manually manipulating the "Text to Columns" feature combined with complex restructuring steps. Instead, we will leverage three specialized functions--TEXTSPLIT, TRANSPOSE, and VSTACK--to automate this normalization process completely. The ultimate goal is to convert a wide, horizontal list of values into a neatly stacked, consolidated column, instantly ready for immediate data processing. The visual transformation we aim to achieve is represented in the image below, illustrating how aggregated names move from one cell to multiple rows:



	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		**Players**			
9		Andy			
10		Bob			
11		Chad			
12		Doug			
13		Eric			
14		Frank			
15		Greg			
16		Henry			
17		Ian			
18					
19					
20					

The introduction of **Dynamic array formulas** has revolutionized data handling, turning previously complex tasks into simple, formula-driven operations. The following sections provide the necessary context, preparation steps, and the precise formulas required to execute this powerful and efficient data manipulation successfully.

Preparation: Structuring the Source Data

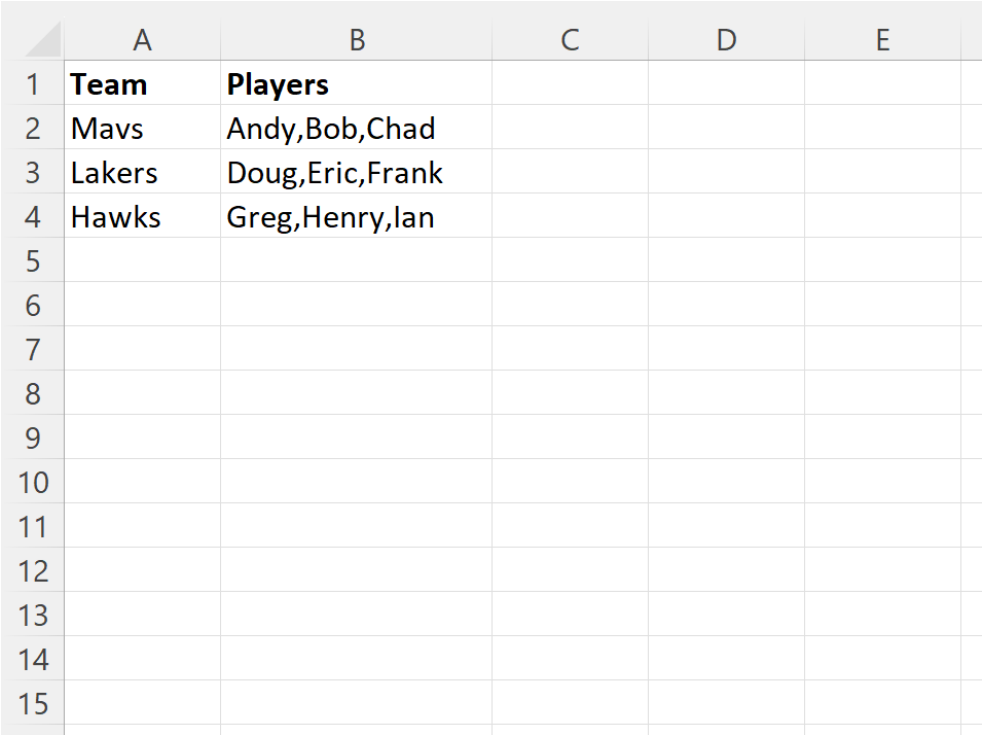
Before deploying any dynamic array functions, the first and most crucial phase involves correctly organizing your initial data structure. For the purpose of this tutorial, we will utilize a sample dataset that accurately simulates common real-world data scenarios, where a primary, consistent identifier (such as a Team Name) is logically associated with a cell containing multiple related entries (Player Names) separated by the comma delimiter.

To begin, accurately enter the data into your spreadsheet. It is vital to ensure the source data is clearly segmented, featuring one distinct column for the categorical variable (e.g., Team) and a second column dedicated to the cell containing the comma-separated values (e.g., Players). This initial, clear setup forms the foundation necessary for the successful application of the subsequent splitting and restructuring functions, ensuring accuracy throughout the process.

For our demonstration, we will use a small dataset detailing basketball teams and their respective players. Please input the following structure into your worksheet, ideally starting in cell A1 to maintain clear referencing:

Column A: **Team Name** (The Identifier)
Column B: **Comma-Separated Player Names** (The Aggregated Data)

This systematic input allows us to precisely and efficiently target the necessary range when we implement the core splitting mechanism in the next phase. The input structure of your dataset should resemble the following visualization:



	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					

Grasping this structure is key to understanding the objective. Our aim is to take the multi-value entries housed in Column B (specifically cells B2, B3, B4, and so on) and expand them vertically, ensuring each new row retains the corresponding Team Name from Column A. This process of data normalization is essential for generating a clean, row-based output that is perfectly structured for relational analysis and reporting.

Step 1: Splitting Values Horizontally using TEXTSPLIT

The primary mechanism responsible for isolating and separating the aggregated values is the modern TEXTSPLIT function. This powerful function, introduced in recent Excel versions, is purpose-built to take a text string and divide it based on specified delimiters, outputting the results as a **dynamic array** that automatically "spills" across the required number of columns. Crucially,

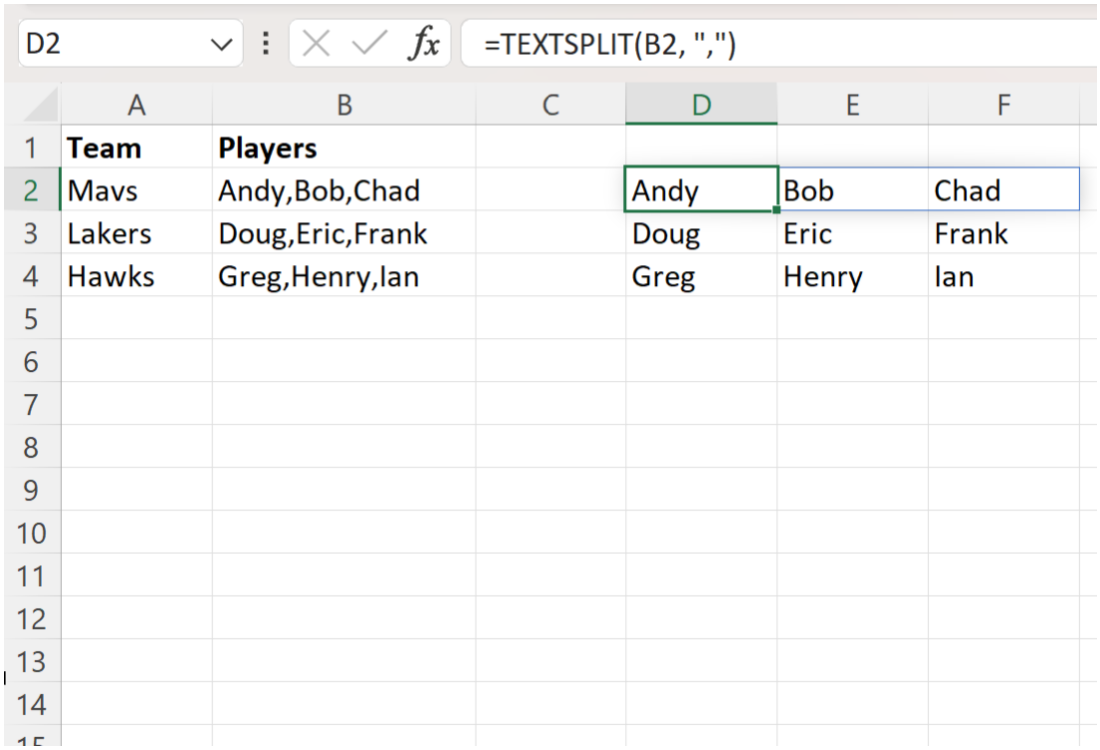
this dynamic capability completely eliminates the need for legacy, static tools like "Text to Columns," which often require data to be pasted as values, thus breaking formulaic dependencies.

The general syntax of the function is `=TEXTSPLIT(text, col_delimiter, , , , )`. For our specific objective, we only need to utilize the first two mandatory arguments: the cell containing the source text and the column delimiter (in this case, the comma). The inherent benefit of this function lies in its dynamic output capacity; the size of the resulting array automatically adjusts itself based on the number of individual items found within the source cell, making it highly adaptable to varying data lengths.

To implement this initial step, navigate to cell **D2** (or the first empty cell immediately adjacent to your source data) and input the following concise formula. This instruction directs Excel to examine the content of cell **B2** and create a split wherever a comma (",") appears:

=TEXTSPLIT(B2, ",")

Once you press Enter, the results will instantaneously "spill" horizontally into cells D2, E2, F2, and so forth. Because this is a true Dynamic array formula, you do not need to manually drag it horizontally. After verifying the successful split, click and drag the formula down from cell D2 to cover the remaining rows corresponding to your entire dataset (D3, D4, etc.). This action applies the crucial splitting logic to all subsequent rows in Column B, creating a temporary block of data where the individual player names are now distributed across columns D, E, and F, as demonstrated in the visual below. At this point, we have successfully separated the comma-separated values into distinct columns.



It is important to recognize that while the data is now split, the output remains horizontally oriented and grouped by the original row entry (the Team). Our ultimate objective, however, requires these values to be stacked vertically into a single, continuous column. This essential requirement necessitates the next transformation step: transposition.

Step 2: Reorienting Data for Stacking with TRANSPOSE

While the previous step successfully isolated the values across columns, the data is still segmented horizontally by its original team grouping. To prepare this intermediate data structure for final vertical consolidation, we must employ the TRANSPOSE function. The core function of **TRANSPOSE** is to change the orientation of a range or array, converting rows into columns and columns into rows. This matrix manipulation is absolutely essential because the final stacking function, VSTACK, performs its task most efficiently when appending arrays that are already structured vertically, one directly after the other.

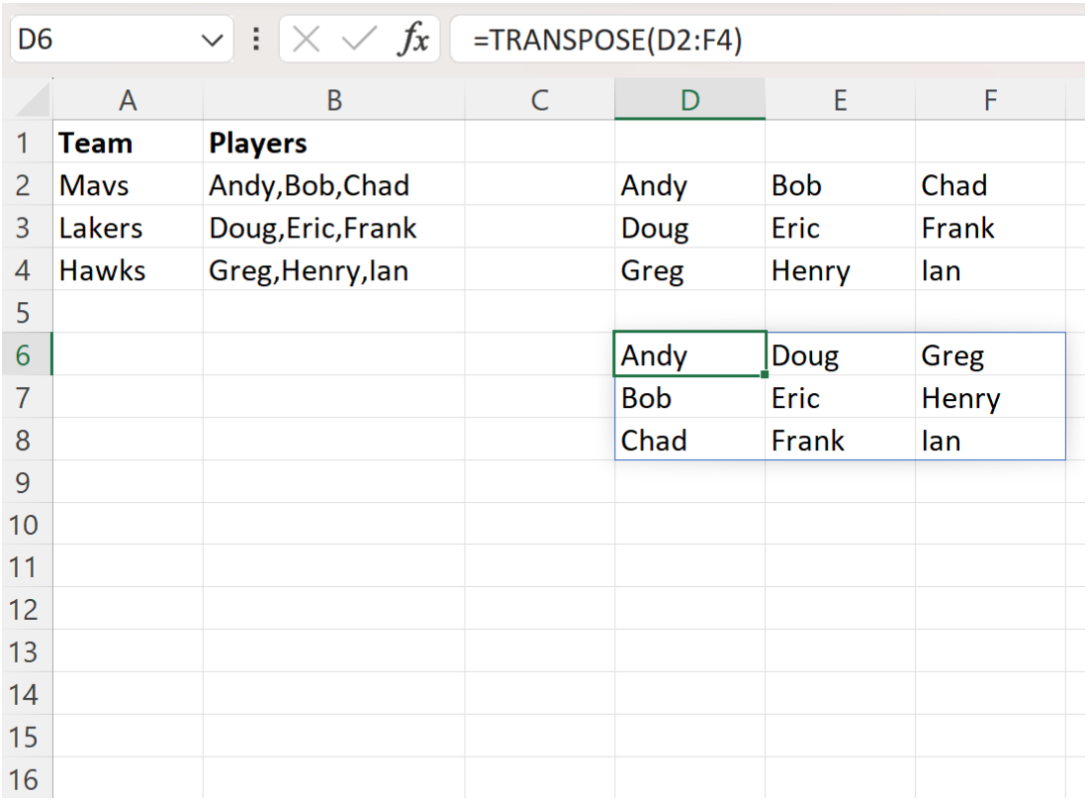
Our task now is to transpose the entire block of split player names that we generated in the previous step (the range D2:F4 in our specific example). By transposing this entire block, we effectively convert the team-specific horizontal arrays (e.g., D2, E2, F2, representing the players of Team 1) into corresponding vertical arrays (e.g., D6, D7, D8). This vertical structure makes them perfectly stackable for the final consolidation phase.

Locate an empty area of your worksheet, such as cell **D6**, and enter the following formula. It is

critically important that the range **D2:F4** accurately and fully encompasses all the cells generated by the TEXTSPLIT function in the preceding step. If your dataset contains more teams or more players per team, adjust the range reference accordingly to capture all the intermediate data:

=TRANSPOSE(D2:F4)

Executing this formula will immediately result in three distinct vertical arrays, which begin spilling in cells D6, E6, and F6, respectively. Each of these new vertical arrays corresponds to the players from a single original team, now listed sequentially, one below the other. For instance, the original horizontal row D2:F2 is now reorganized vertically in the range D6:D8. This critical transformation successfully prepares the data for vertical stacking, as illustrated by the resulting table structure below:



	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	Doug	Greg
7				Bob	Eric	Henry
8				Chad	Frank	Ian
9						
10						
11						
12						
13						
14						
15						
16						

This intermediate transposition step ensures that when we proceed to the stacking phase, Excel correctly stacks the entire list of players from Team 1, followed sequentially by the entire list of players from Team 2, and so forth, maintaining the logical data grouping required for a clean final output. Attempting to skip this transposition and stack the horizontal arrays directly would inevitably lead to an incorrect, fragmented, and unusable data arrangement.

Step 3: Vertical Consolidation using VSTACK

The final and most defining step in this data restructuring pipeline is the consolidation of all the individual, vertically-oriented arrays into one continuous, unified column. For this task, we rely on the VSTACK function (Vertical STACK), which stands as the final component of Excel's modern Dynamic array formula capabilities. This function is designed explicitly to append multiple arrays or ranges one below the other in a sequential manner.

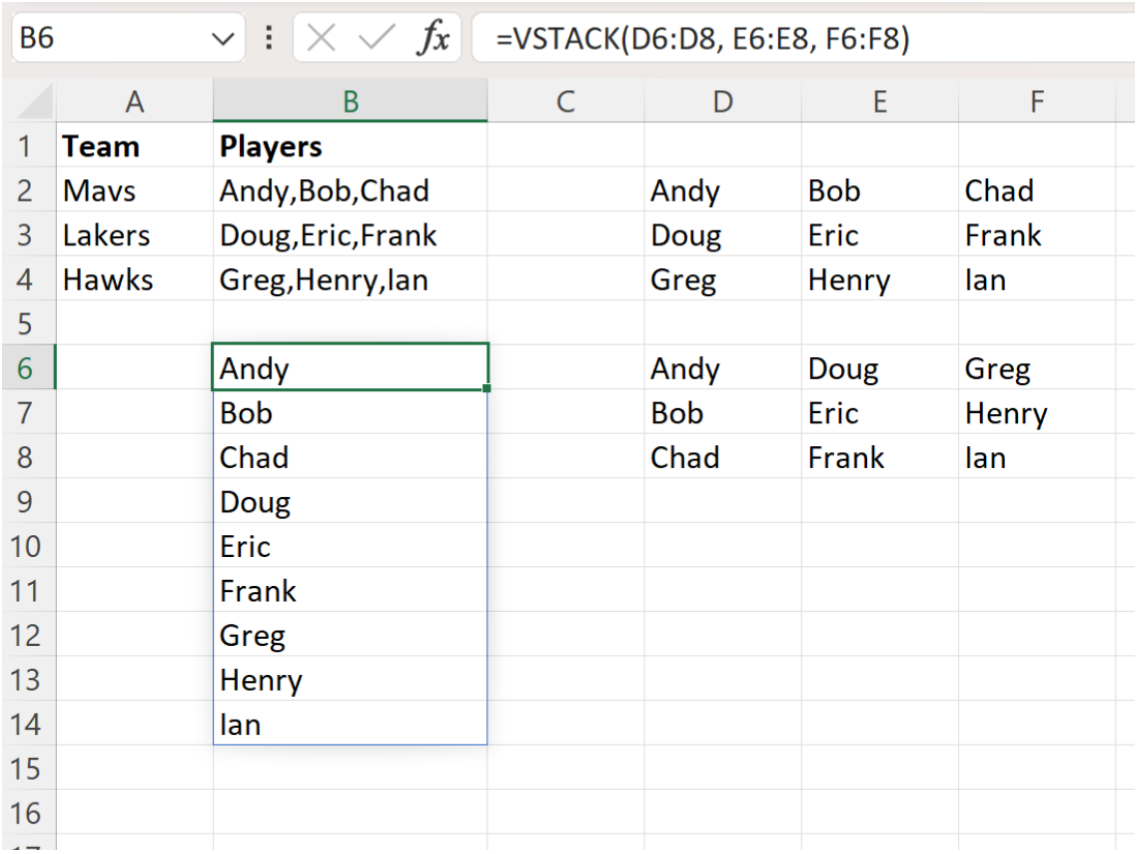
To use VSTACK, we simply need to reference the three distinct vertical arrays that were created in Step 2 (D6:D8, E6:E8, and F6:F8) as separate arguments within the function. Because these arrays were carefully derived using the TRANSPOSE function, they possess the perfect vertical structure for seamless consolidation.

In cell **B6** (or any desired final output location that you ensure does not overlap with existing data), input the following formula. Double-check that the ranges accurately reflect the output cells generated during your preceding TRANSPOSE step:

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

This single command instructs Excel to take the first array (D6:D8) and stack it vertically, then immediately append the second array (E6:E8) directly underneath it, followed by the third array (F6:F8). The immediate result is a single, contiguous column containing all player names, successfully achieving the objective of splitting the original comma-separated values into individual, analyzable rows.

The resulting output clearly showcases the successful transformation, where the data has been efficiently normalized from a wide, aggregated format to a long, clean format, rendering it substantially more suitable for subsequent data analysis and detailed reporting:



	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						

By implementing this methodology, we have successfully utilized a concise combination of modern dynamic array functions to convert a complex, aggregated dataset into a clean, row-level structure. This technique offers significant efficiency gains, particularly when managing large datasets, as it completely eliminates the need for error-prone manual copy-pasting or reliance on intricate VBA scripting, favoring instead concise, powerful, and easily maintainable formulas.

Conclusion: Mastering Modern Data Transformation

The streamlined technique detailed in this guide--the strategic combination of **TEXTSPLIT**, **TRANSPOSE**, and **VSTACK**--represents the most sophisticated and efficient methodology available today for tackling complex data restructuring challenges within Excel. This process expertly harnesses the full potential of Dynamic array formulas, enabling intermediate results to automatically spill without the tedious requirement of explicit range definition, thereby greatly simplifying formula construction and ensuring long-term maintenance is straightforward.

By achieving mastery over these three specialized functions, you acquire the capability to rapidly clean, normalize, and prepare data that is frequently imported in a less-than-ideal aggregated format. Remember that the key to executing this pipeline flawlessly is a clear understanding of the specific role each function plays in the sequence:

TEXTSPLIT's role is to accurately isolate individual values from the delimited string and output them horizontally across columns.
TRANSPOSE's role is to reorient these resulting horizontal arrays into stackable vertical arrays, preparing them perfectly for consolidation.
VSTACK's role is to sequentially combine these prepared vertical arrays into one final, comprehensive, and clean list.

This powerful three-step process is a fundamental skill for advanced data cleaning and preparation in any modern Excel environment, allowing analysts to move quickly from raw, aggregated input to normalized, analyzable data.

Additional Resources

For users seeking to expand their knowledge of data manipulation techniques, particularly those involving dynamic arrays and advanced text processing, the official Microsoft Support website provides exhaustive documentation. You can find complete documentation for the **VSTACK** function, alongside other essential array manipulators, on these authoritative resources.

The following tutorials explain how to perform other common operations in Excel, leveraging similar modern techniques: