

How to Flatten Data in Excel with Transpose

Authored by
stats writer

February 1, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Flatten Data in Excel with Transpose*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128940>

The concept of **data flattening** in Excel is fundamental to effective data management and analysis. Essentially, flattening involves taking data organized across multiple rows and columns (often a two-dimensional array) and reorganizing it into a streamlined, single column or row format. This transformation is crucial for ensuring that your dataset meets the requirements for advanced analysis techniques, such as generating automated reports, performing statistical calculations, or utilizing modern Pivot Tables and visualizations.

Historically, achieving this rearrangement often relied on manual methods or the older **Transpose** function, which swaps rows and columns but is limited in its ability to truly unstack complex, multi-dimensional ranges into a single continuous stream. While the Transpose function remains useful for simple restructuring, true data normalization often requires condensing multiple fields into a single variable list, which is the primary objective of modern flattening techniques. When data is flattened, it becomes significantly easier to manipulate, summarize, and visualize large datasets effectively, setting the stage for more powerful data insights.

This guide will explore the definitive, modern approach to flattening data in Excel using dynamic array functions, ensuring optimal structure for subsequent data analysis and reporting.

Flatten Data in Excel (With Example)

The Modern Approach: Utilizing the TOCOL Function

For users with a modern version of Excel (Microsoft 365 or Excel 2021 and later), the most efficient and straightforward method for data flattening is the introduction of the **TOCOL** function. This powerful dynamic array function is specifically designed to take a range or array of values and return a vertical array--or a single column--of those values. This eliminates the need for complex helper columns, array formulas requiring Ctrl+Shift+Enter, or lengthy VBA scripts, thus simplifying the workflow dramatically.

The core advantage of **TOCOL** is its inherent simplicity. It processes the input array row by row (the default behavior), pulling all values sequentially and stacking them into one tall column. This reorganization aligns perfectly with database normalization principles, making the data ready for use in other functions or features that demand a single-column input, such as sorting, unique counting, or filtering.

You can use the **TOCOL** function in Excel to flatten an array into a single column. This function is part of the new suite of array manipulation tools that streamline complex data restructuring tasks.

Syntax and Arguments of the TOCOL Function

Understanding the syntax of **TOCOL** is essential for unlocking its full potential. The function accepts up to three arguments, allowing for significant control over which values are included in the final flattened output, particularly concerning blanks and error values within the source range.

The general syntax is: `=TOCOL(array, ,)`

The arguments break down as follows:

array (Required): This is the range or array of values that you wish to flatten. This is typically a multi-row, multi-column selection of your source data.

(Optional): This numerical argument controls how the function handles blank cells and error values within the input array. This is a powerful feature for cleaning data during the flattening process.

0 (Default): Keep all values (including blanks and errors).

1: Ignore blanks only.

2: Ignore errors only.

3: Ignore both blanks and errors.

(Optional): This logical argument determines the order in which the array is read.

FALSE or Omitted (Default): Scan the array by **row** (left-to-right, then top-to-bottom).

TRUE: Scan the array by **column** (top-to-bottom, then left-to-right).

For a basic, sequential flattening operation where you want to keep all data points, the simplest form is sufficient, requiring only the `array` argument. For example, to convert the array of values in the range **B2:E4** into a single vertical column, the formula is concise and direct:

=TOCOL(B2:E4)

This approach ensures that the output dynamically spills into the required number of rows, handling all the complexity internally.

Practical Example: Flattening Quarterly Sales Data

To illustrate the power and simplicity of the **TOCOL** function, let us consider a common business scenario: consolidating sales data organized by year and quarter. When data is presented in a wide, cross-tabular format, it can be difficult to run uniform calculations across all periods simultaneously. Flattening this data transforms it into a long-format dataset ready for immediate analysis.

Suppose we have the following dataset in Excel that displays the total sales made by some

company during each quarter of three consecutive years:

| | A | B | C | D | E | F |
|----|-------------|-----------|-----------|-----------|-----------|---|
| 1 | Year | Q1 | Q2 | Q3 | Q4 | |
| 2 | 2021 | 4 | 8 | 9 | 8 | |
| 3 | 2022 | 14 | 7 | 17 | 22 | |
| 4 | 2023 | 9 | 12 | 15 | 23 | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |

This table spans from cell B2 to E4, where the rows represent the years and the columns represent the quarters. Suppose we would like to flatten this table into a single continuous column, starting with the Q1 2021 sales value and ending with the Q4 2023 sales value.

To perform this transformation, we select an empty cell--for instance, cell **A6**--and input the necessary **TOCOL** formula, referencing the entire data range that needs consolidation. Since we want to preserve all values, we omit the optional `ignore` argument.

We can type the following formula into cell **A6** to do so:

=TOCOL(B2:E4)

The following screenshot shows how to use this formula in practice, demonstrating the resulting dynamic array spill:

| | A | B | C | D | E | F |
|----|-------------|-----------|-----------|-----------|-----------|---|
| 1 | Year | Q1 | Q2 | Q3 | Q4 | |
| 2 | 2021 | 4 | 8 | 9 | 8 | |
| 3 | 2022 | 14 | 7 | 17 | 22 | |
| 4 | 2023 | 9 | 12 | 15 | 23 | |
| 5 | | | | | | |
| 6 | 4 | | | | | |
| 7 | 8 | | | | | |
| 8 | 9 | | | | | |
| 9 | 8 | | | | | |
| 10 | 14 | | | | | |
| 11 | 7 | | | | | |
| 12 | 17 | | | | | |
| 13 | 22 | | | | | |
| 14 | 9 | | | | | |
| 15 | 12 | | | | | |
| 16 | 15 | | | | | |
| 17 | 23 | | | | | |
| 18 | | | | | | |

Interpreting the Flattened Output Sequence

The result of applying the **TOCOL** function is a clean, vertical list of all 12 sales values. It is critical to understand the order in which these values are pulled, especially when the `scan_by_column` argument is left at its default setting (scanning by row).

When scanning by row, Excel processes all cells in the first row (B2:E2) before moving to the second row (B3:E3), and so forth. This ensures a logical, sequential arrangement of the data based on the original structure.

We can see that the formula has successfully flattened the values from the table into a single column that shows the sales values for each quarter and year in chronological order.

The sequence proceeds row by row, ensuring that the sales figures are preserved in their original chronological context:

The first value in the column shows the sales for **Q1 2021** (Cell B2).

The second value in the column shows the sales for **Q2 2021** (Cell C2).

The third value in the column shows the sales for **Q3 2021** (Cell D2).

The fourth value in the column shows the sales for **Q4 2021** (Cell E2).

The process then repeats for the subsequent years (Row 3, 2022, and Row 4, 2023), and so on.

Advanced Flattening: Handling Blanks and Errors

One of the most valuable aspects of using the **TOCOL** function, especially in large, messy datasets, is its optional second argument, `ignore`. This argument allows users to clean the data during the flattening process itself, preventing issues that might arise from feeding blank cells or error values into downstream calculations.

For instance, if your original sales data contained some missing quarterly data represented by blank cells, these blanks would normally be included as zeros (or blank strings) in the final output, potentially skewing aggregates. By setting the `ignore` argument to 1, we instruct **TOCOL** to skip these empty cells entirely. The result is a dense, clean column containing only meaningful numerical values.

Similarly, if you encounter `#N/A` or `#DIV/0!` errors within your data range, using an `ignore` argument of 2 will remove only the errors, while 3 removes both blanks and errors. This selective exclusion capability is a significant efficiency improvement over traditional methods that required complex filtering steps after the data was transposed or stacked.

Alternative Methods for Data Restructuring

While **TOCOL** is the preferred method for modern Excel users, especially for quick array transformations, it is important to acknowledge alternative techniques, particularly for users of older Excel versions or when handling exceptionally large, external datasets.

One powerful alternative is Power Query (also known as Get & Transform Data), which is built into Excel. Power Query excels at importing, transforming, and reshaping data from various sources. Within Power Query's editor, the process of flattening wide data into a tall format is called "Unpivoting." Unpivoting allows you to select key identifier columns and then automatically stack all remaining column headers and their corresponding values into two new columns--an attribute column and a value column.

Compared to **TOCOL**, Power Query is more robust for complex data cleaning tasks and datasets exceeding Excel's row limits, as it handles the transformation outside the standard Excel grid. However, **TOCOL** offers instant, volatile recalculation directly on the worksheet, making it faster for internal, smaller worksheet manipulations.

Why Flattening Data is Essential for Advanced Analysis

The primary motivation behind data flattening is achieving a state of "tidy data," a crucial concept in

statistical computing and data science. Tidy data adheres to three rules: each variable forms a column, each observation forms a row, and each type of observational unit forms a table. Wide data formats (like the original quarterly sales table) violate the "each variable forms a column" rule because the quarter (Q1, Q2, Q3, Q4) is actually a variable attribute, not a fixed measure.

Once data is flattened into a tall format using **TOCOL**, it is optimized for almost every analytical tool within Excel. Features like [Pivot Tables](#), conditional formatting, data validation based on unique lists, and advanced filtering all perform significantly better when operating on a single, uniform column of values rather than a scattered range.

Furthermore, feeding flattened data into external statistical software or visualization tools (like Power BI or Tableau) becomes seamless, as these applications are standardized to handle long-format datasets. By leveraging the **TOCOL** function, you are not just rearranging numbers; you are preparing your data for deeper, more reliable statistical scrutiny.

Note: You can find the complete documentation for the **TOCOL** function in [Excel](#) on the Microsoft Support website, where detailed examples covering the optional arguments are provided.

Further Tutorials on Excel Data Manipulation

Mastering data flattening is just one step in becoming proficient with Excel's dynamic array capabilities. To continue expanding your expertise in data preparation and complex worksheet operations, consider exploring related functions and techniques that complement **TOCOL**.

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

Understanding the **TOROW** function (the horizontal counterpart to **TOCOL**).

Using **VSTACK** and **HSTACK** for combining arrays without restructuring.

Techniques for dynamic sorting and filtering using the **SORT** and **FILTER** functions.