Learning to Filter Cells by Strikethrough Formatting in Microsoft Excel

Authored by

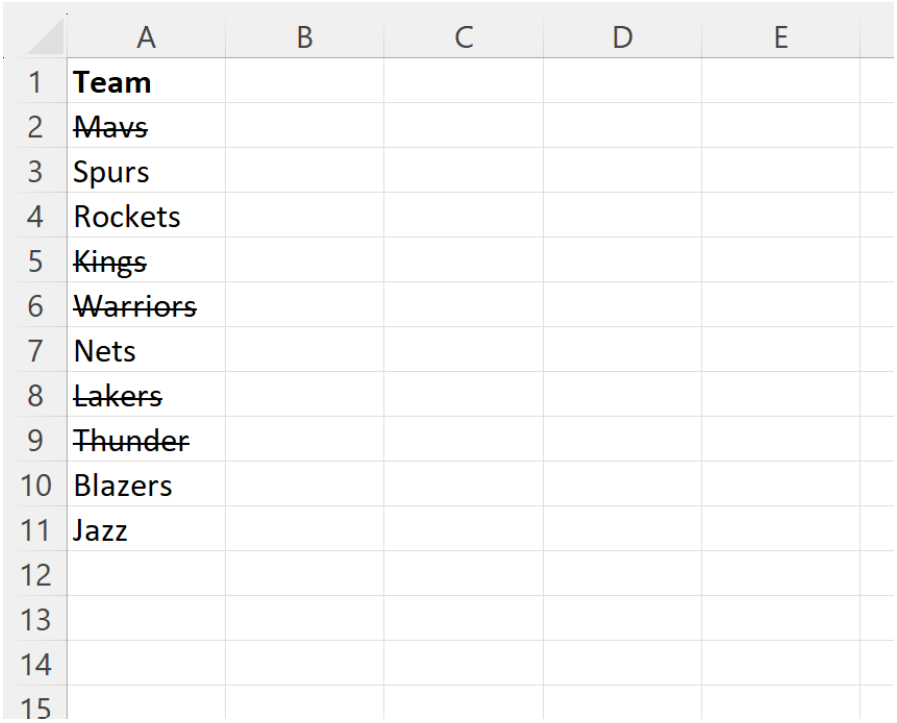
Mohammed looti

November 13, 2025

Introduction: Bridging the Gap Between Formatting and Filtering in Excel

In the dynamic and highly functional environment of Microsoft Excel, users routinely employ visual formatting to categorize and mark data. The application of a strikethrough, for example, is a widely adopted practice used to signify that an item is completed, obsolete, or otherwise irrelevant to current operations. Whether managing a complex project schedule or maintaining a detailed inventory list, the ability to quickly isolate these marked entries is crucial for effective data management and informed decision-making. However, a significant operational challenge arises because Excel's native filtering tools--while excellent for filtering by values, colors, or icons--lack a built-in feature to directly filter based on character formatting like strikethrough. This limitation forces users managing extensive datasets to manually scan thousands of rows, leading to inefficiency and potential errors.

This guide provides a precise, powerful solution to overcome this inherent limitation. We will demonstrate a proven methodology that leverages the customization capabilities of Visual Basic for Applications (VBA) to create a custom function. This function effectively translates the visual attribute of strikethrough formatting into a standard, filterable data point (a **Boolean** TRUE/FALSE value). By integrating this simple piece of code, we transform a visual queue into a quantifiable metric that Excel's powerful filtering engine can process. The following steps will meticulously walk you through preparing your data, activating the necessary development tools, writing the custom code, and finally, applying the native filter to achieve precise isolation of strikethrough entries, dramatically enhancing your productivity and data accuracy within Excel.



	A	B	C	D	E
1	**Team**				
2	~~Mavs~~				
3	Spurs				
4	Rockets				
5	~~Kings~~				
6	~~Warriors~~				
7	Nets				
8	~~Lakers~~				
9	~~Thunder~~				
10	Blazers				
11	Jazz				
12					
13					
14					
15					

Step 1: Prepare Your Data in Excel

The foundation of any successful data manipulation or filtering operation in Excel begins with properly structured data. Before implementing the custom filtering solution, it is essential to ensure your dataset is neatly organized in contiguous columns and rows, typically starting from cell **A1**. This structure ensures seamless interaction with the custom VBA function that will be deployed later. For the purpose of this tutorial, we recommend using a sample dataset that accurately reflects your real-world application, containing both entries formatted with a strikethrough and those without, allowing for a clear demonstration of the filtering effectiveness.

Open your target Excel workbook and confirm that the data you intend to filter is correctly entered and that the strikethrough formatting has been applied precisely where needed. This visual indicator is the trigger our custom function will detect. It is vital to recognize that the filtering mechanism relies entirely on the character formatting applied to the cells. Any errors in the source data or inconsistencies in applying the strikethrough will naturally affect the filtering output.

We must also plan for the addition of a new, crucial element: the helper column. This column, which will be placed immediately adjacent to your primary data (e.g., Column B if your data is in Column A), will house the results of our custom function. It is this helper column that converts the visual formatting into explicit, quantifiable data points (TRUE/FALSE) that Excel's native filtering tools require. By dedicating this space, we establish a robust basis for the subsequent steps involving code execution and advanced macros and filtering techniques.

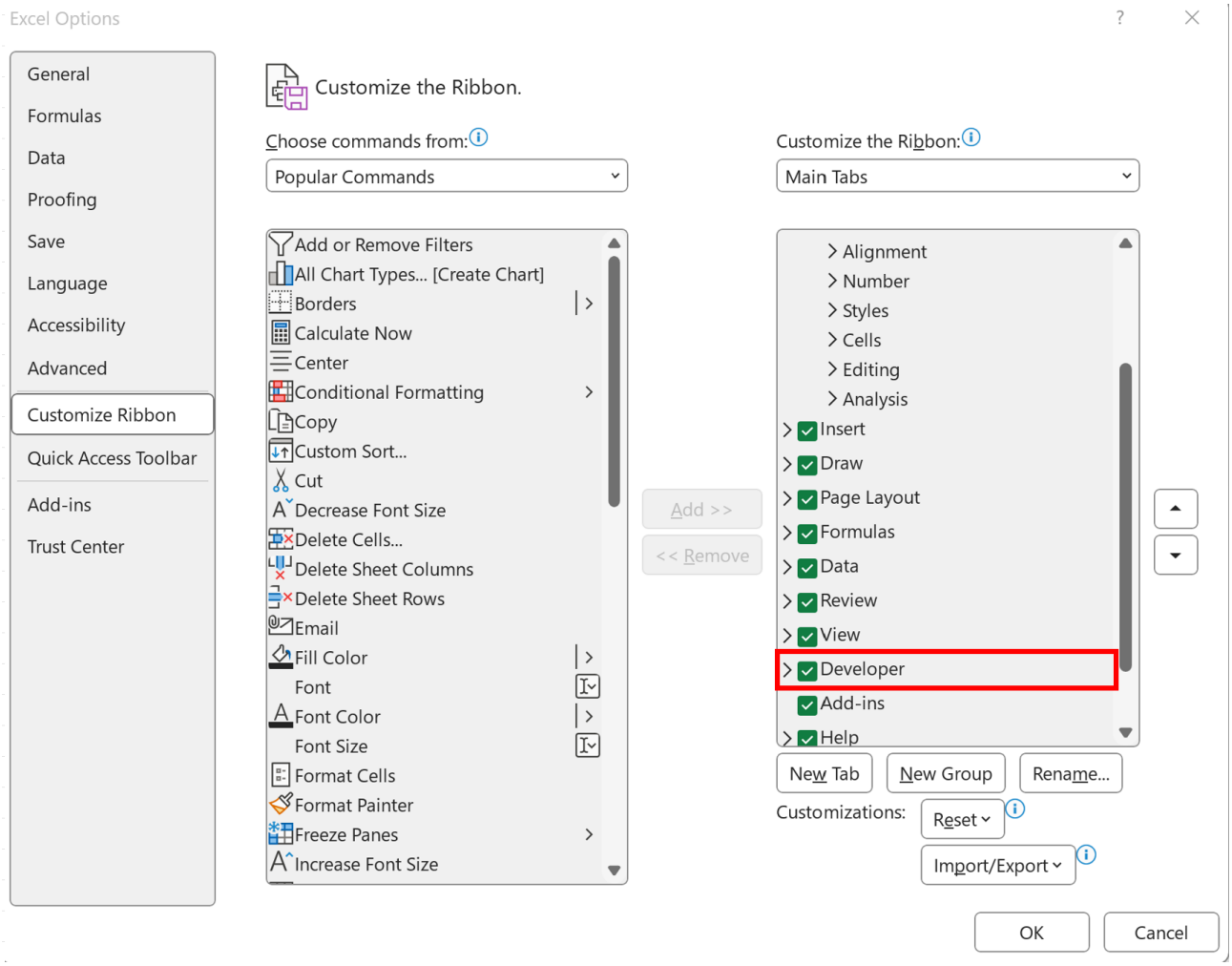
	A	B	C	D	E
1	**Team**				
2	~~Mavs~~				
3	Spurs				
4	Rockets				
5	~~Kings~~				
6	~~Warriors~~				
7	Nets				
8	~~Lakers~~				
9	~~Thunder~~				
10	Blazers				
11	Jazz				
12					
13					
14					
15					

Step 2: Enable the Developer Tab in Excel

To implement any custom code, such as the VBA function we require, access to Excel's developer tools is mandatory. These tools are centralized within the **Developer** tab on the Excel Ribbon. Since this tab is intentionally hidden in default installations to keep the user interface clean, our second step focuses on making this critical gateway visible. The Developer tab grants access to the Visual Basic Editor (VBE), where all custom macros and functions are written and managed.

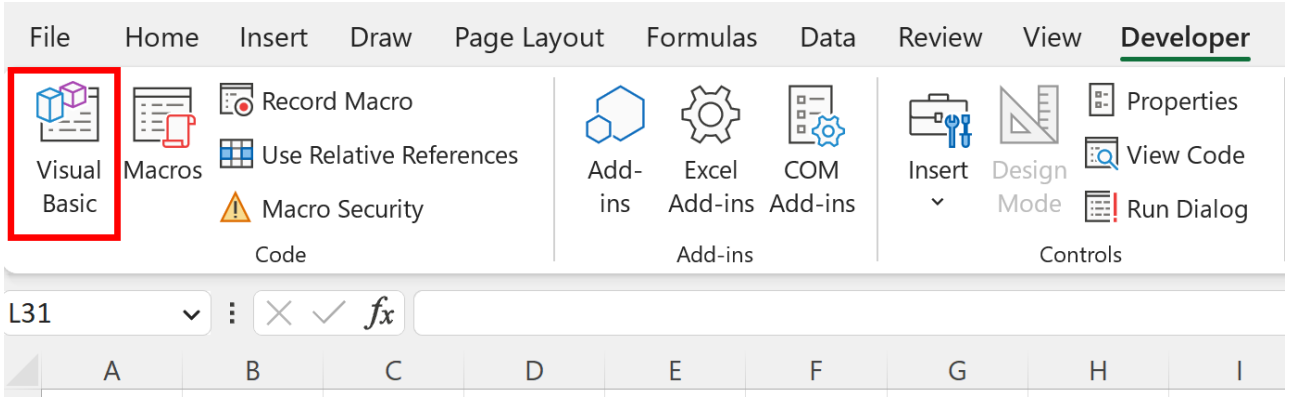
The process for enabling the **Developer** tab is simple and uniform across most recent versions of Excel. Start by navigating to the **File** tab located in the top-left corner of the application window. Clicking **File** opens the Backstage view. From the options presented on the left-hand navigation pane, select **Options** to launch the comprehensive Excel Options dialog box. This centralized control panel provides extensive customization possibilities for the application environment.

Within the Excel Options dialog, locate and click on the **Customize Ribbon** category in the left pane. On the right side, under the section dedicated to customizing the main tabs, you will see a list of available Ribbon tabs. Scroll down this list until you find the entry labeled **Developer**. It is essential to ensure that the checkbox next to this tab name is marked. Once you have confirmed the selection, click the **OK** button at the bottom of the dialog box to save your changes and exit. Upon returning to your worksheet, the **Developer** tab will now be prominently displayed alongside your other main tabs, ready for the next step of code deployment.

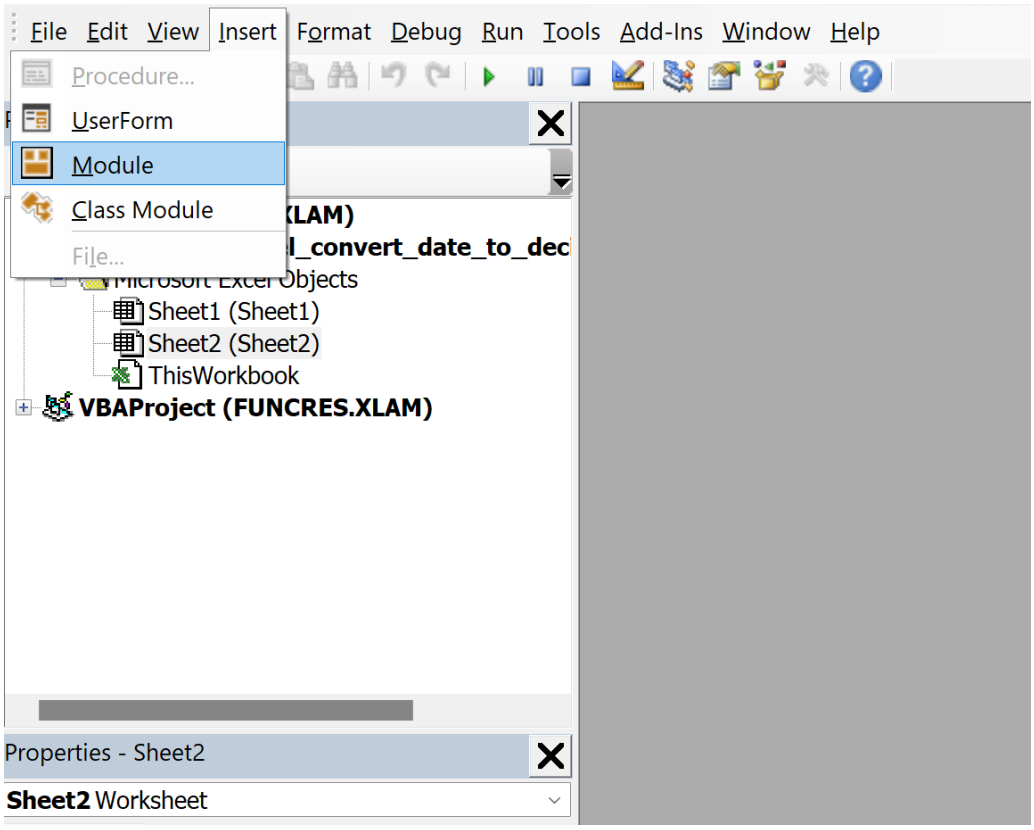


Step 3: Develop a Custom Function Using VBA

With the necessary tools now accessible, we move to the core of the solution: defining a custom function capable of detecting strikethrough formatting. This function, which we will name `FindStrikethrough`, utilizes the powerful capabilities of VBA to inspect a cell's properties and return a simple **Boolean** indicator. To begin, click on the newly enabled **Developer** tab and select **Visual Basic** from the "Code" group, which will launch the Visual Basic Editor (VBE) environment.



Inside the VBE, custom functions that are intended to be used directly in worksheet formulas must reside within a standard module, ensuring they are globally available across the workbook. To insert a new module, navigate to the VBE menu bar, click on **Insert**, and then select **Module**. A new, blank code pane will appear on the right side of the VBE window. This is where you will paste the code that defines our custom function.

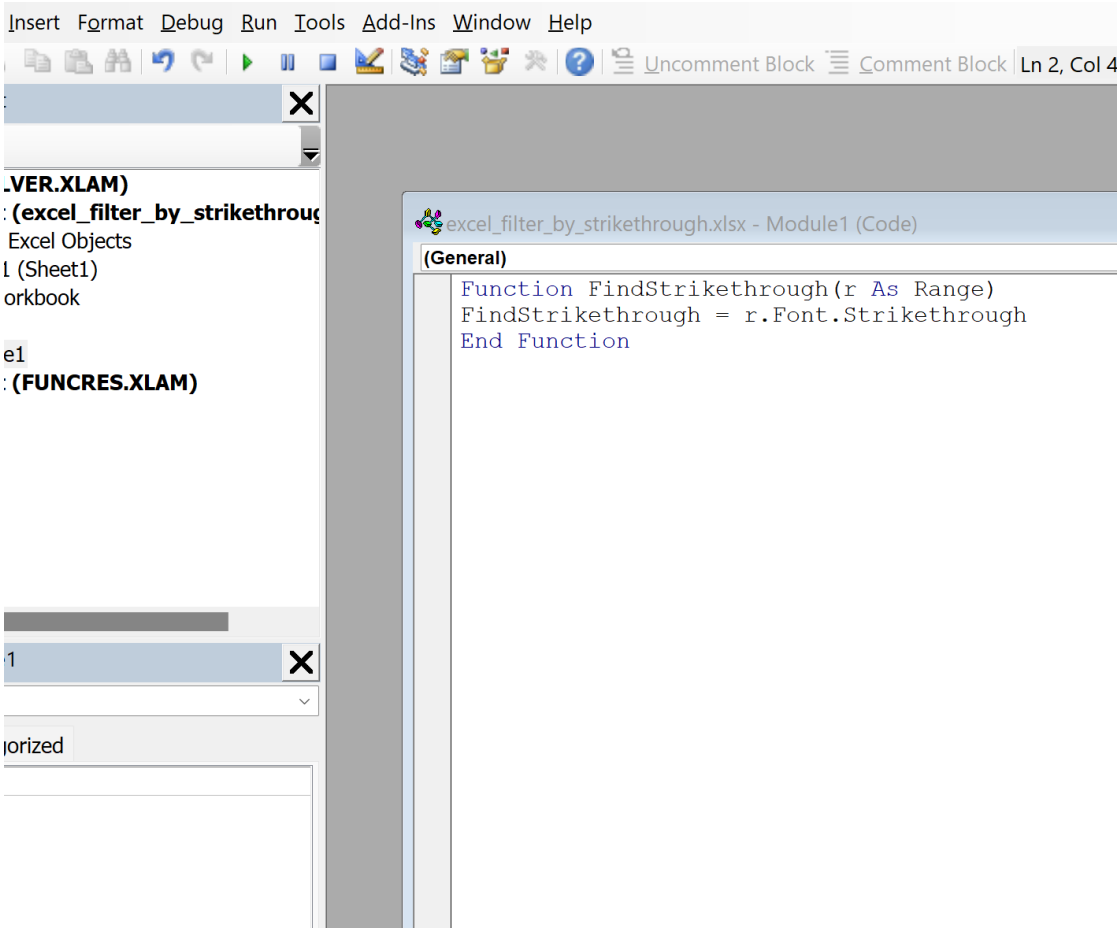


The following snippet defines the `FindStrikethrough` function. It takes one argument, `r`, which represents the cell range being evaluated. The core logic relies on checking the `r.Font.Strikethrough` property. If the formatting is present, this property returns `TRUE`;

otherwise, it returns `FALSE`. This elegant function is lightweight and performs precisely the transformation we need, converting a format into a value.

```
Function FindStrikethrough(r As Range)
FindStrikethrough = r.Font.Strikethrough
End Function
```

After meticulously pasting the code into the module editor, review it for accuracy, paying close attention to syntax and capitalization. Once the code is correctly inserted, you do not need to explicitly save the module; the code is automatically associated with the workbook. You may simply close the VBE window. The custom function is now ready to be called like any standard macro or formula directly within your Excel worksheet.



Step 4: Integrate the Custom Function into the Worksheet

The creation of the `FindStrikethrough` function in VBA is only the first part of the integration process. The next crucial step involves applying this custom logic directly to your data range using

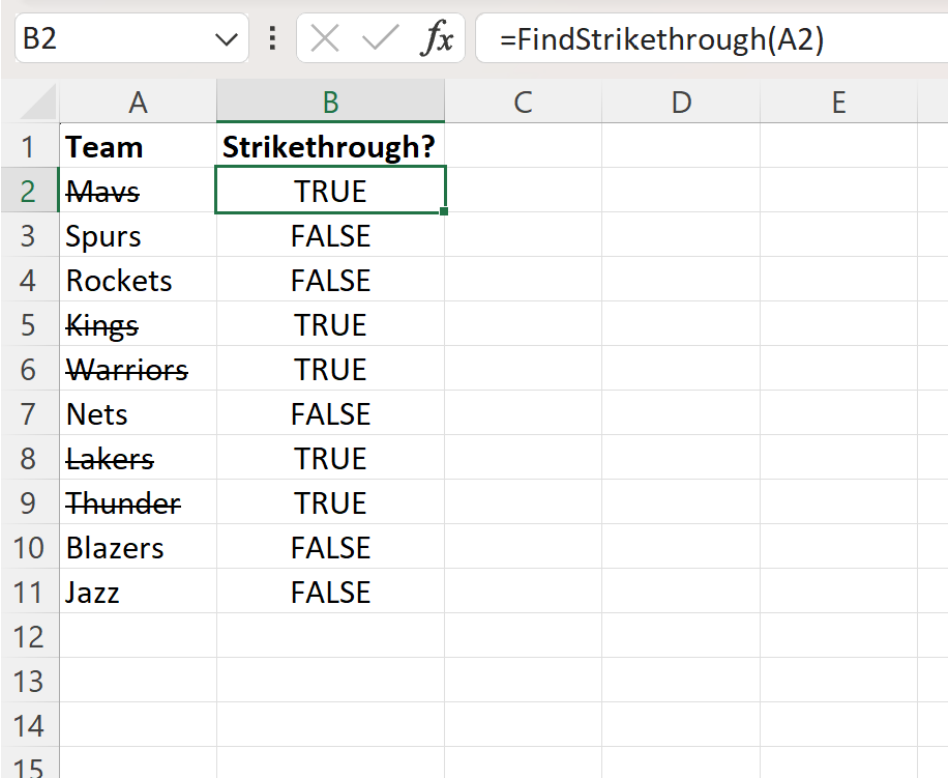
a standard Excel formula. This action generates the helper column (Column B) that is essential for native filtering, effectively making the previously hidden formatting visible as explicit data.

Start by selecting the first empty cell in your designated helper column. Assuming your data begins in Column A and your header row is Row 1, this cell would be **B2**. In cell **B2**, enter the following formula: `=FindStrikethrough(A2)`. This formula instructs Excel to execute your custom function, passing the corresponding cell in the primary data column (A2) as the argument. The function then returns a Boolean output--`TRUE` if cell A2 has a strikethrough, and `FALSE` if it does not.

=FindStrikethrough(A2)

After entering the formula and pressing Enter, the cell **B2** will display the accurate status. To scale this operation across your entire dataset, utilize Excel's efficient fill handle feature. Click back onto cell **B2**, locate the small, solid square in the bottom-right corner, and click and drag this handle downwards until you cover all rows corresponding to your data in Column A. This action automatically adjusts the cell range reference in the formula (e.g., B3 will contain `=FindStrikethrough(A3)`, and so on). Column B will now be fully populated with clear `TRUE` or `FALSE` flags for every entry, providing a quantifiable basis for filtering.

It is important to understand that this helper column is the linchpin of our solution. It transforms the otherwise inaccessible visual formatting into a data type that is universally recognized by Excel's data processing tools. This crucial step successfully converts a formatting attribute into a standard, filterable data point, setting the stage for the final filtering operation.



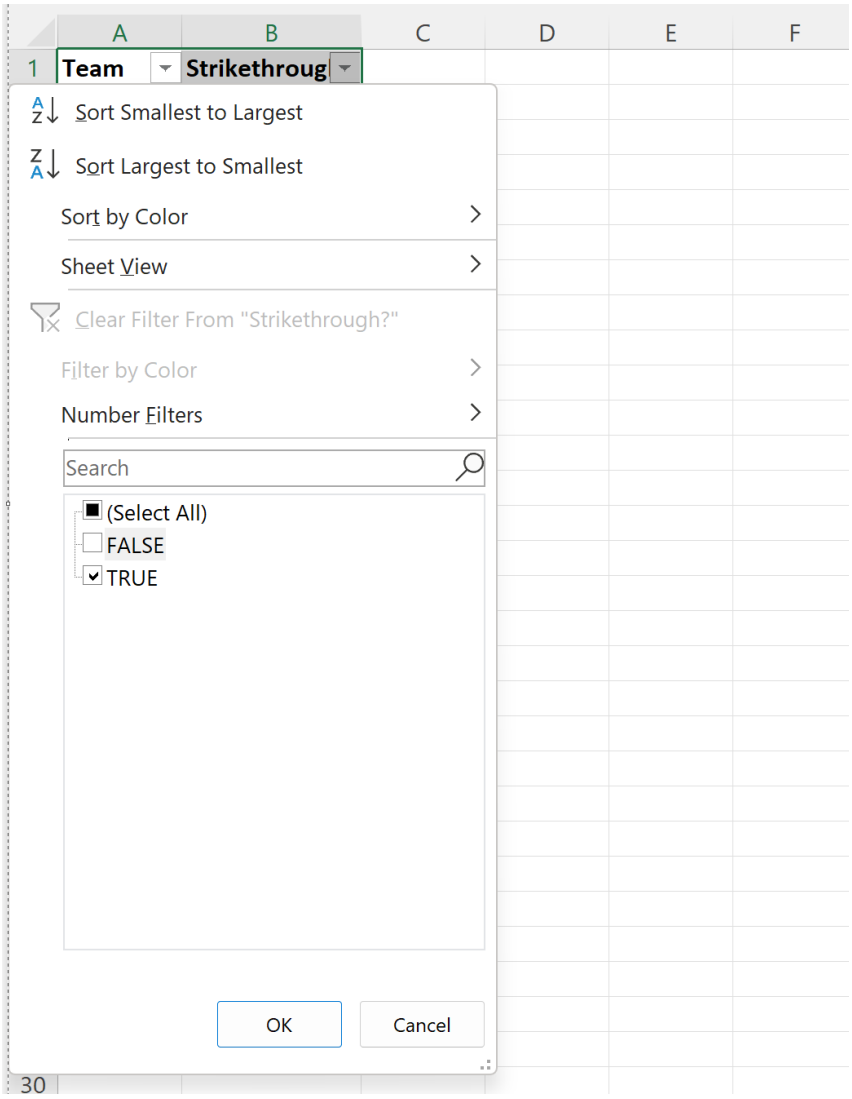
	A	B	C	D	E
1	**Team**	**Strikethrough?**			
2	~~Mavs~~	TRUE			
3	Spurs	FALSE			
4	Rockets	FALSE			
5	~~Kings~~	TRUE			
6	~~Warriors~~	TRUE			
7	Nets	FALSE			
8	~~Lakers~~	TRUE			
9	~~Thunder~~	TRUE			
10	Blazers	FALSE			
11	Jazz	FALSE			
12					
13					
14					
15					

Step 5: Apply Excel's Native Filter to Isolate Strikethrough Entries

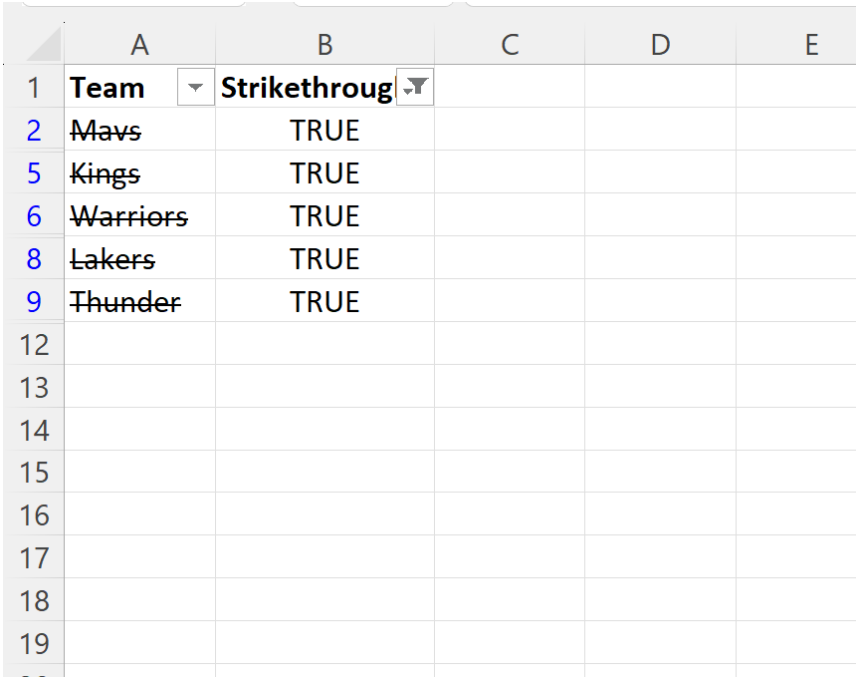
With the helper column (Column B) now explicitly defining the strikethrough status of every data point using `TRUE` or `FALSE` values, the final step involves applying Excel's standard filter mechanism. This culmination allows for instantaneous isolation of the desired entries, efficiently fulfilling the original goal of filtering by formatting.

Begin by selecting the entire data range, which must include both the original data (e.g., Column A) and the newly created helper column (Column B). Next, navigate to the **Data** tab located on the Excel Ribbon. Within the "Sort & Filter" group, click the **Filter** icon. This action applies dropdown arrows to the header row of your selected range, activating the interactive filtering functionality across all columns.

To isolate the strikethrough entries, click the dropdown arrow located in the header of the helper column (Column B). The filter menu will display the unique values present: `TRUE` and `FALSE`. Since `TRUE` indicates that a strikethrough is present, you must uncheck the box next to **FALSE**. This instructs Excel to temporarily hide all rows where the custom function returned a negative result.



Once you click **OK**, the worksheet will refresh instantly, displaying only those data rows that have strikethrough formatting applied in the monitored column. This provides a clean, focused view, ideal for reporting, auditing, or performing bulk actions on completed or obsolete tasks. The flexibility of this approach is significant: by simply toggling the filter selection (e.g., choosing to display only `FALSE` entries), you can instantly isolate all tasks that still require attention, offering a dynamic and powerful tool for data analysis and workflow management within Excel.



	A	B	C	D	E
1	**Team**	**Strikethroug**			
2	~~Mavs~~	TRUE			
5	~~Kings~~	TRUE			
6	~~Warriors~~	TRUE			
8	~~Lakers~~	TRUE			
9	~~Thunder~~	TRUE			
12					
13					
14					
15					
16					
17					
18					
19					

Conclusion and Advanced Excel Proficiency

Successfully filtering by strikethrough formatting demonstrates the immense power unlocked by integrating VBA custom functions with Excel's native features. This technique transforms visual cues into actionable data points, enabling precise control over complex datasets. Mastering Excel proficiency involves recognizing where standard features fall short and applying targeted automation solutions, like the custom macro demonstrated here.

To further enhance your data management and analytical capabilities, we encourage continuous exploration of Excel's advanced functionalities. Expanding your knowledge beyond basic filtering to include dynamic array formulas, complex conditional formatting rules, and advanced VBA scripting allows for the automation of repetitive tasks and the handling of ever more intricate data challenges.

Consider delving into the following related areas to build upon the foundation established in this guide. These resources cover essential skills necessary for transitioning from a basic user to an advanced data analyst capable of leveraging Excel's full spectrum of tools for maximum efficiency and accuracy:

Exploring advanced conditional formatting techniques that dynamically change cell appearance based on complex rules.

Developing robust data validation rules and managing large-scale dropdown lists for input control.

Utilizing structured references (Tables) for easier formula maintenance and data integrity.

Writing more complex macros using the VBE to automate multi-step processes.

Implementing techniques for auditing and debugging VBA code to ensure reliability and performance.