Learn to Filter Excel Cells by Font Style: A Guide to Isolating Bold Text

Authored by

Mohammed looti

November 13, 2025

In advanced data management using Microsoft Excel, analysts often encounter situations where data isolation must be based on cell attributes rather than raw values. A frequent and complex requirement is to filter records based on whether the cell content has been styled with a **bold font**. Because Excel's native filtering tools are exclusively designed to evaluate cell contents (numbers, dates, and text), they cannot directly recognize formatting properties. This limitation necessitates a technical workaround to bridge the gap between visual styling and logical data processing.

Consider a scenario involving a substantial dataset where key entries, such as high-priority tasks or critical financial figures, have been manually emphasized by applying bold formatting. Our primary objective is to efficiently isolate and view only these critical rows. The following image illustrates a typical data sample where some entries are formatted in **bold font**, and others are not. Our goal is to develop a reliable mechanism to display only the bolded rows.

	A	B	C	D	E	F
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						
16						

The most effective and robust solution for this sophisticated filtering task is the deployment of custom logic built using **Visual Basic for Applications (VBA)**. By creating a specialized custom function, commonly referred to as a User Defined Function (UDF), we can programmatically query the formatting characteristics of any cell. The output of this function--a simple TRUE or FALSE--can then be used as input for Excel's standard data filtering tools, providing a clean, effective, and reliable filter based solely on font style. This guide details the step-by-step process required to implement this powerful solution.

Step 1: Data Structuring and Prerequisites Check

Before initiating the technical implementation, it is vital to ensure that your source data is correctly set up and that the target formatting--the **bold font**--is accurately applied across the required range. While the underlying principles of this method are universally applicable, the process is best demonstrated using a clearly structured data range. Verify that the cells intended for filtering based on bold formatting are correctly styled before proceeding to the VBA environment.

Using the data structure shown below as our ongoing illustration, observe that the dataset contains a mixture of bold and standard text entries. The core challenge we address is establishing a programmatic method capable of distinguishing between these two distinct visual styles efficiently. The success of the subsequent filtering steps depends entirely on the accuracy of the formatting established in this initial phase.

	A	B	C	D	E	F
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						
16						

A crucial prerequisite is confirming the visibility of the Developer Tab within your Excel ribbon interface. This tab serves as the gateway to the Visual Basic Editor (VBE), which is the dedicated environment for writing and managing custom functions and automation scripts. If you have not previously worked with advanced Excel features, the Developer Tab may be hidden by default. The following step provides comprehensive instructions for enabling this essential tool.

Step 2: Activating the Developer Tab for Custom Functionality

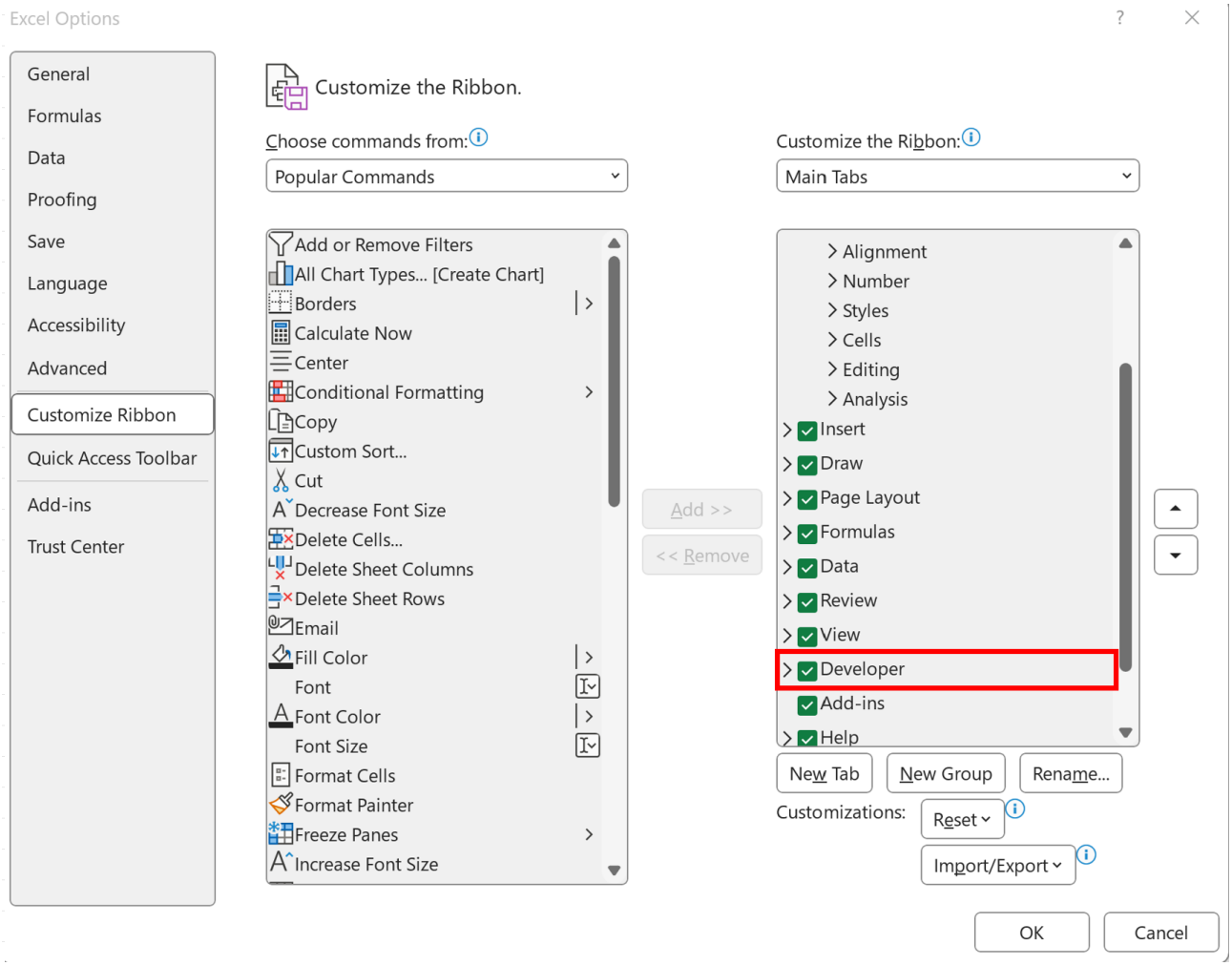
To gain access to the tools necessary for creating and managing custom macros and functions in

Excel, the **Developer Tab** must be explicitly enabled. This component, which grants access to the Visual Basic Editor, is typically disabled upon initial installation. Enabling it is a simple configuration change executed through the standard Excel Options menu.

Execute the following precise sequence of steps to make the **Developer Tab** visible and ready for use:

Click the **File** tab, located in the upper-left corner of the Excel application ribbon.
Select **Options** from the menu displayed near the bottom of the File view, which launches the main Excel Options configuration window.
In the Options window's left navigation pane, choose the **Customize Ribbon** category.
Examine the list labeled **Main Tabs** on the right side of the window. Scroll until you locate the entry titled **Developer**.
Ensure that the checkbox adjacent to the **Developer** entry is checked (selected).
Click **OK** to confirm your selection, save the changes, and close the options dialog box.

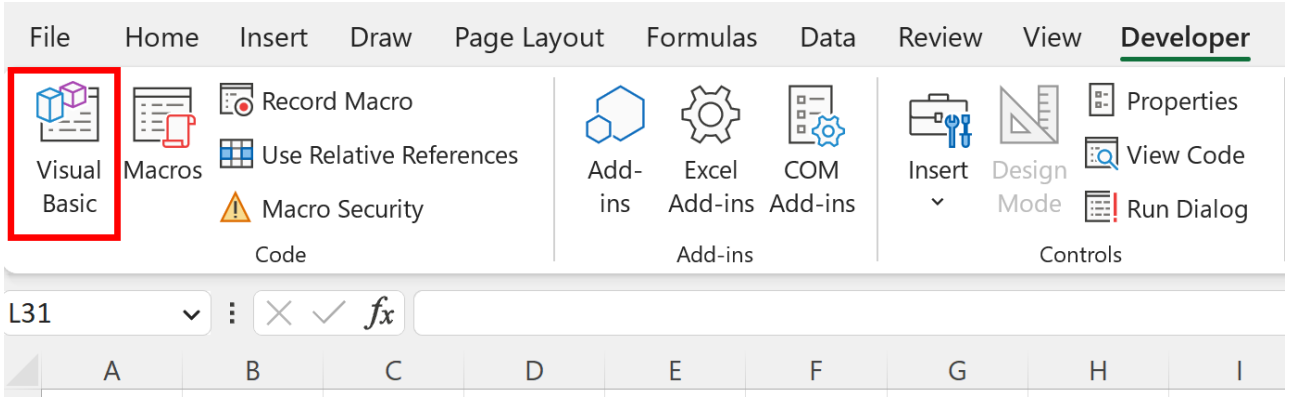
Upon successful completion of these steps, the **Developer Tab** will now be prominently displayed on your main Excel ribbon. This activation is fundamental, as it unlocks the advanced programming tools necessary to implement the custom formatting detection logic using **VBA**.



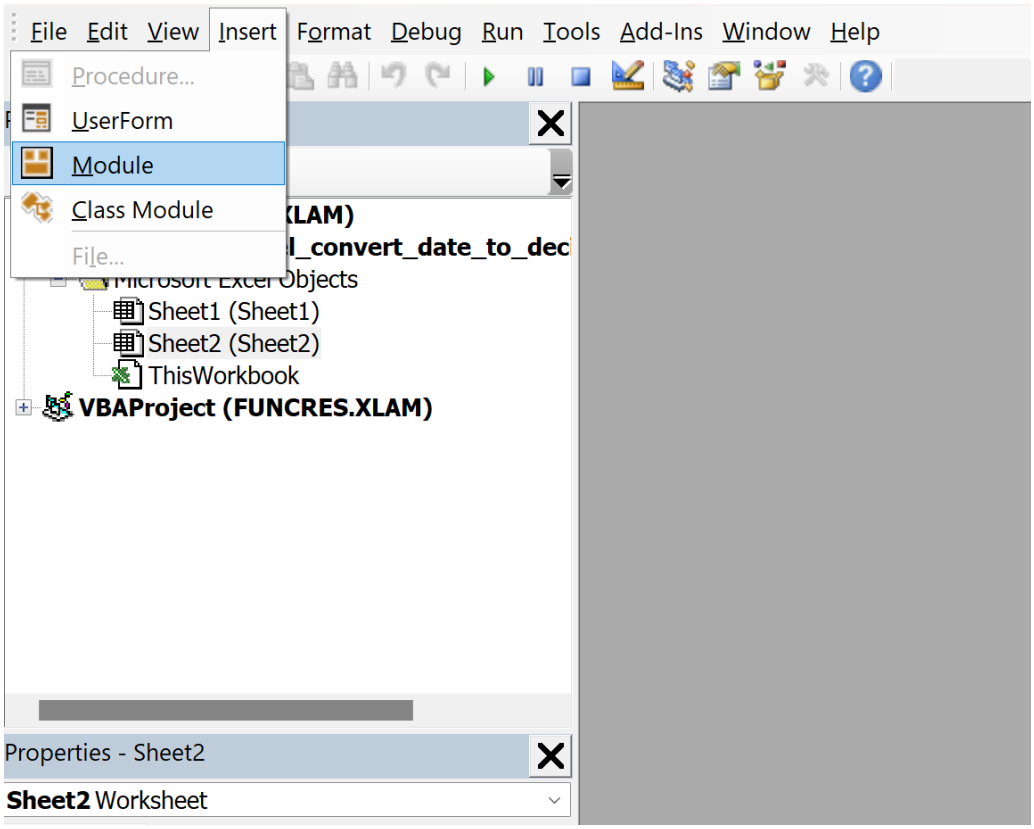
Step 3: Creating the VBA User Defined Function (UDF)

The core technical component of this solution is a small, specialized script written in **VBA**. This script is designed to function as a custom formula, meaning it can be called directly from any cell within your worksheet, just like built-in functions such as SUM or VLOOKUP. Its sole purpose is to inspect the formatting properties of a specified cell and return a logical value indicating the status of the bold property.

To begin coding, click the newly activated **Developer** tab, and then select the **Visual Basic** icon, usually found in the far left of the ribbon. This action launches the Visual Basic Editor (VBE), which is the integrated development environment for VBA projects.



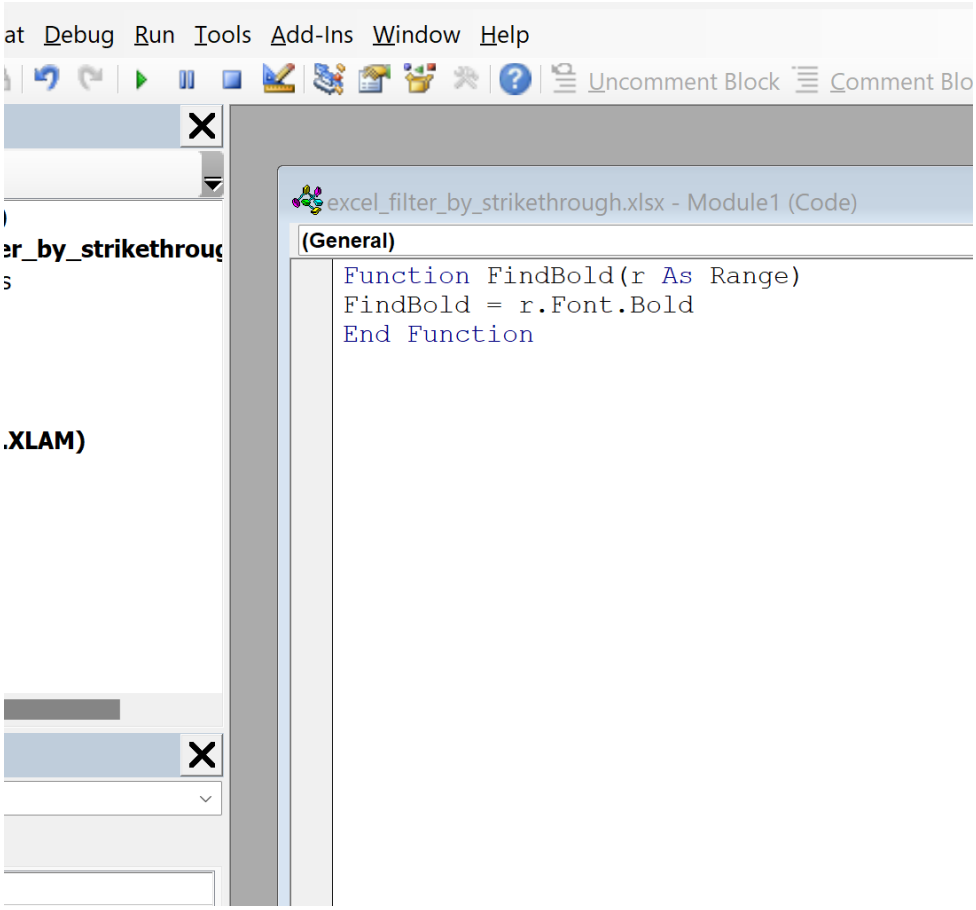
Within the VBE interface, the custom function must be inserted into a standard module. Navigate to the **Insert** menu within the VBE and choose **Module**. This will open a new, blank code editor window. Placing the code in a standard module ensures that the resulting macro is available for use across all sheets within the entire workbook.



Carefully paste the following function definition into the opened module window. This concise code defines a function named `FindBold` that takes a range object (`r`) as its input argument and returns the state of the `.Font.Bold` property. This property inherently yields a Boolean result (either TRUE or FALSE):

```
Function FindBold(r As Range)
FindBold = r.Font.Bold
End Function
```

Once the code has been correctly input and verified, the function is immediately active and ready for deployment within your worksheet. You may close the VB Editor; the custom function is automatically saved within the current workbook environment and is now accessible just like any native Excel formula.



Step 4: Implementing the Helper Column for Logical Conversion

The next critical phase involves utilizing the new `FindBold` function across your dataset to establish a helper column. This column performs the vital task of translating the visual formatting (whether the text is bold or not) into explicit, logical data values (TRUE or FALSE). This conversion is essential because it provides the standardized input required for Excel's built-in filter functionality.

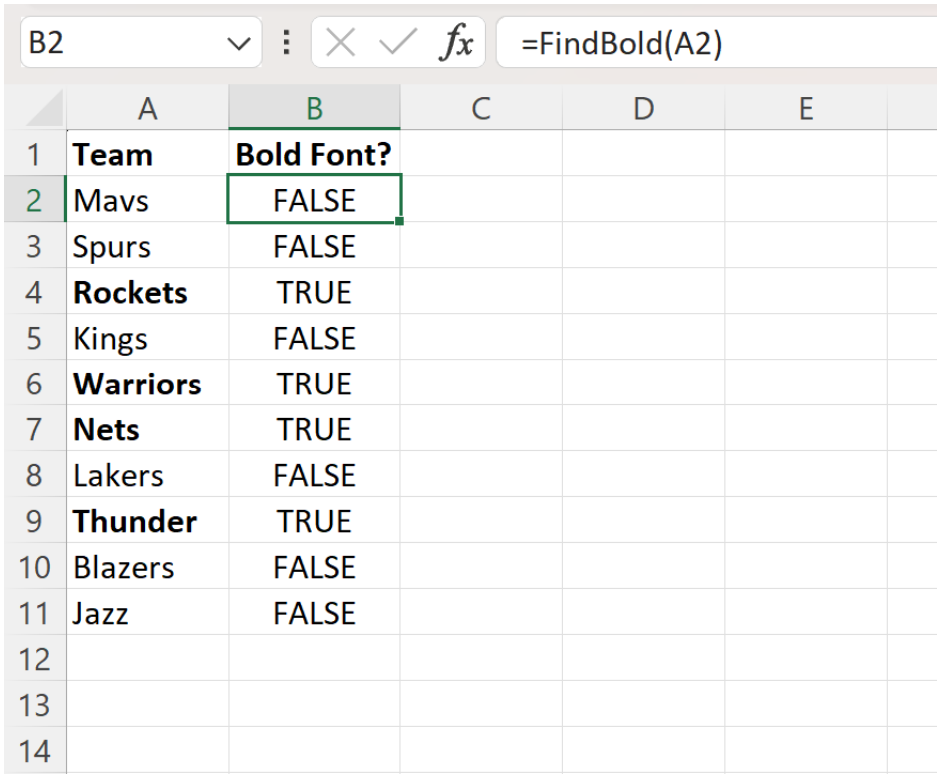
Select the cell adjacent to the first data point you intend to check (for example, cell **B2**, assuming

your primary data starts in Column A). Enter the following formula, ensuring it references the corresponding data cell (**A2**):

=FindBold(A2)

The function executes instantly, returning the text **TRUE** if the cell A2 is formatted as bold, and **FALSE** if it is not. This output is recognized natively as a Boolean data type by Excel, which is perfect for filtering.

To apply this check across your entire dataset, simply use the fill handle (the small green square at the bottom-right corner of the active cell **B2**) and drag the formula down to the last row of your data table. Column B will now act as a dynamic indicator, providing logical status for the bold formatting of every entry in the monitored column. This helper column successfully transforms a visual property into quantifiable data, making the subsequent filtering step possible.



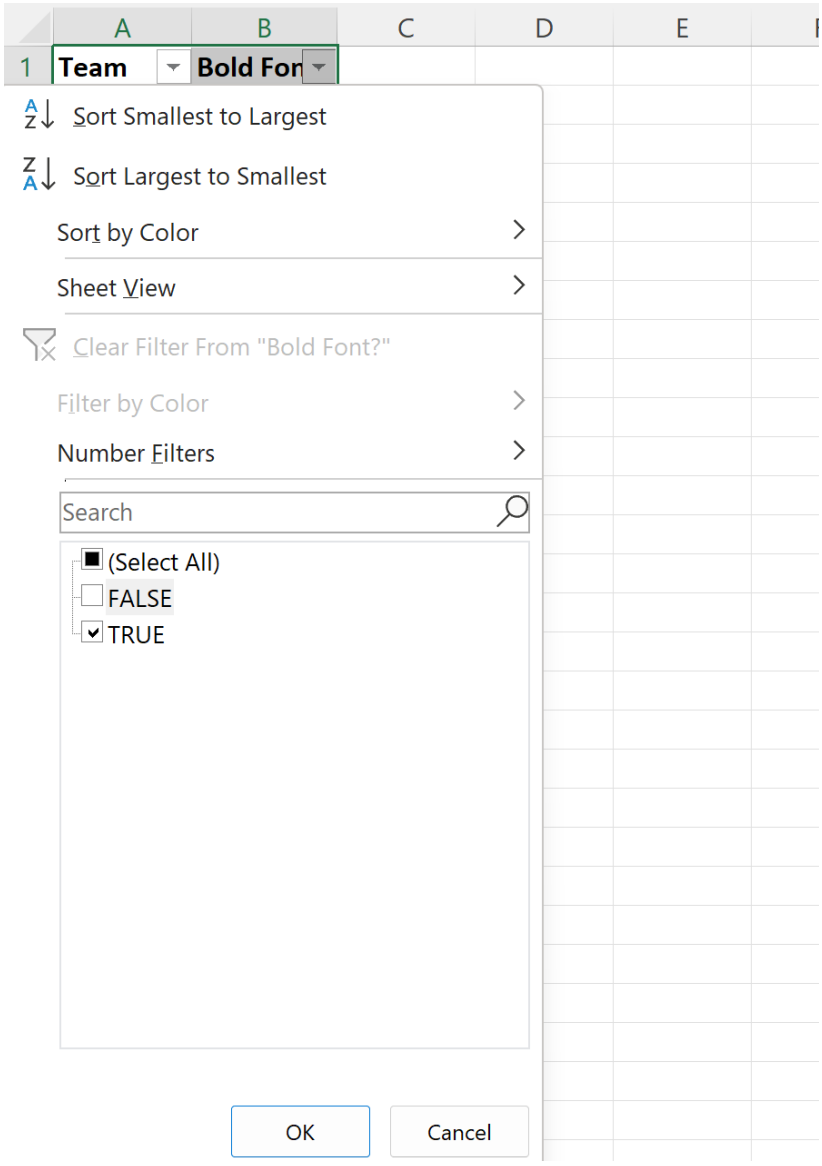
	A	B	C	D	E
1	**Team**	**Bold Font?**			
2	Mavs	FALSE			
3	Spurs	FALSE			
4	**Rockets**	TRUE			
5	Kings	FALSE			
6	**Warriors**	TRUE			
7	**Nets**	TRUE			
8	Lakers	FALSE			
9	**Thunder**	TRUE			
10	Blazers	FALSE			
11	Jazz	FALSE			
12					
13					
14					

Step 5: Executing the Format-Based Filter

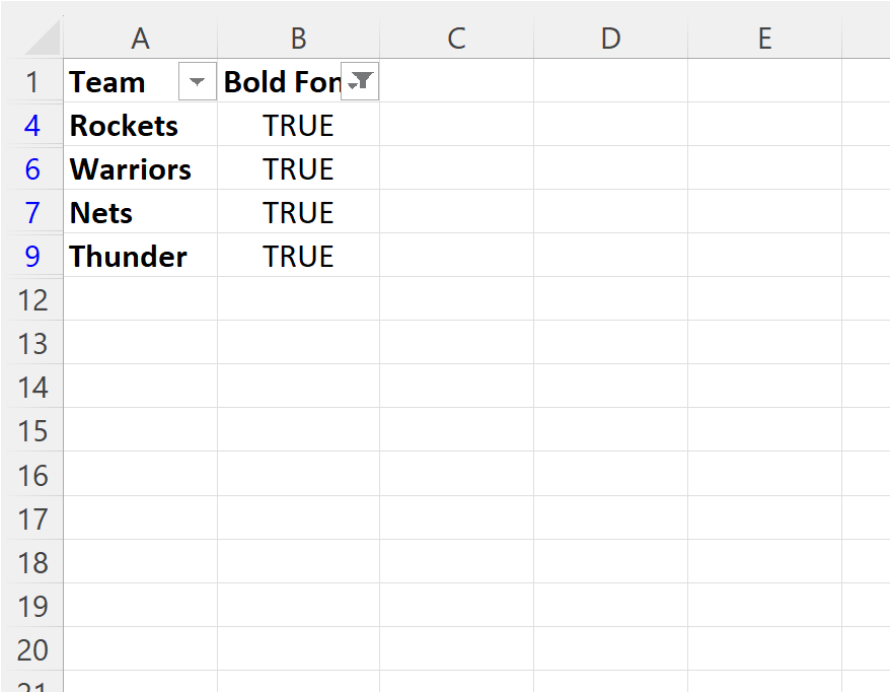
Once the helper column is populated with TRUE and FALSE values, the final step involves leveraging standard Excel data filtering capabilities. Since the formatting status is now represented by explicit, filterable text values, we can manipulate this column as we would any other data field.

Perform the final filtering operation by following these concise, detailed instructions:

Select the complete data range, ensuring you include all headers and the newly created helper column (e.g., covering the range **A1:B11**).
Navigate to the **Data** tab located on the Excel ribbon interface.
Click the **Filter** icon, which is situated within the **Sort & Filter** group. This action applies filter dropdown arrows to all columns in the header row.
Click the dropdown arrow corresponding to the helper column (e.g., the column you titled "Bold Font?").
In the displayed filter menu, ensure that the checkbox next to **TRUE** is selected, while simultaneously verifying that the box next to **FALSE** is **unchecked**. This configuration instructs Excel to only display rows where the bold status is affirmatively TRUE.



Clicking **OK** will instantaneously refresh the spreadsheet view. Only the rows containing text formatted in **bold font** within the monitored column will remain visible, successfully achieving the desired format-based filter. This sophisticated method demonstrates that although Excel lacks built-in formatting filters, the extensibility provided by **VBA** empowers users to customize functionality to meet even the most advanced data manipulation requirements.



	A	B	C	D	E
1	**Team**	**Bold Fon**			
4	**Rockets**	TRUE			
6	**Warriors**	TRUE			
7	**Nets**	TRUE			
9	**Thunder**	TRUE			
12					
13					
14					
15					
16					
17					
18					
19					
20					

Note on Versatility and Adaptation: The User Defined Function approach is remarkably flexible. Should your requirement change to filtering for all cells that are **not** bolded, the modification is straightforward: you would simply reverse the filtering condition in Step 5, unchecking **TRUE** and checking **FALSE**. Furthermore, this fundamental macro structure can be easily adapted to detect other formatting attributes, such as checking for italics (by changing the VBA code to `r.Font.Italic`) or specific font colors (using `r.Font.Color`).

Additional Resources for Advanced Excel Functionality

Developing the ability to extend Excel's core functionality through custom **VBA** functions is a valuable skill set for any data professional. To enhance your expertise in automating complex tasks and managing data based on properties beyond simple values, we recommend exploring these related concepts:

In-depth documentation detailing the `Range.Font` property and its associated methods within the official Excel Object Model documentation.
Tutorials demonstrating effective practices for creating, managing, and debugging multiple User

Defined Functions within a single workbook environment.

Advanced applications of the **Developer Tab**, including methods for integrating custom forms, interactive buttons, and other controls directly into your spreadsheet interfaces.