# Learning Conditional Formatting in Google Sheets: Applying Rules Based on Multiple Text Values

Authored by

**Mohammed looti**

November 9, 2025

One of the most powerful features available in Google Sheets is the ability to use **conditional formatting** to dynamically change the appearance of cells based on specific criteria. While standard formatting rules handle simple numerical comparisons easily, applying rules based on **multiple text values** often requires a more sophisticated approach. This is where the **custom formula** function becomes indispensable.

Leveraging a custom formula allows users to create complex logical conditions that check a single cell against a list of possible string inputs simultaneously. This tutorial will provide a comprehensive, step-by-step guide demonstrating how to implement this technique effectively, ensuring your data visualization is both accurate and immediate.

## The Power of Custom Formulas in Google Sheets

Standard conditional formatting options are generally limited to single conditions, such as "Text contains," "Greater than," or "Is equal to." When faced with the requirement to highlight a cell if it contains value A, B, or C, chaining multiple independent rules can quickly become cumbersome and inefficient, especially as the list of required values grows or requires frequent updates. Relying on multiple rules also increases the computational load on the spreadsheet.

The definitive solution lies in utilizing the **custom formula** option within the conditional formatting rules panel. This feature allows you to input any valid Sheets formula that is guaranteed to return a Boolean result (TRUE or FALSE). If the formula evaluates to TRUE for a specific cell, the designated formatting is applied; conversely, if it evaluates to FALSE, the cell remains unchanged. This powerful mechanism enables the use of advanced logical functions, including the powerful pattern matching capabilities offered by **REGEXMATCH**.

By defining the complex matching logic externally using a single formula, we centralize the criteria necessary for evaluation. This approach not only streamlines the formatting process but also makes the entire rule set significantly easier to manage, audit, and replicate across different datasets. We will now proceed with a practical example to illustrate precisely how this advanced technique works when filtering data based on multiple positional roles.

## Detailed Example: Applying Conditional Formatting Based on Text Values

To demonstrate the practical application of conditional formatting rules based on multiple text strings, let us consider a sample dataset commonly used in sports analytics. Suppose we are managing a spreadsheet that tracks basketball player statistics, specifically detailing their current position and accumulated points. This is a common requirement for coaches or analysts needing quick visual queues within large data tables.

The primary goal is to instantly highlight players who occupy key roles on the court, such as

starting lineup members or specific specialty positions. This requires selecting all relevant cells where the "Position" column contains specific, pre-defined text strings. The dataset we are working with is displayed below, featuring various players, their assigned positions, and their scores:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Position** | **Points** | | |
| 2 | Starting Point Guard | 22 | | |
| 3 | Backup Point Guard | 19 | | |
| 4 | Starting Shooting Guard | 14 | | |
| 5 | Backup Shooting Guard | 12 | | |
| 6 | Starting Small Forward | 30 | | |
| 7 | Backup Small Forward | 22 | | |
| 8 | Starting Power Forward | 34 | | |
| 9 | Backup Power Forward | 12 | | |
| 10 | Starting Center | 15 | | |
| 11 | Backup Center | 10 | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |

Specifically, we aim to apply a visual highlight to every entry in the **Position** column that matches any of the following critical roles, regardless of case variation:

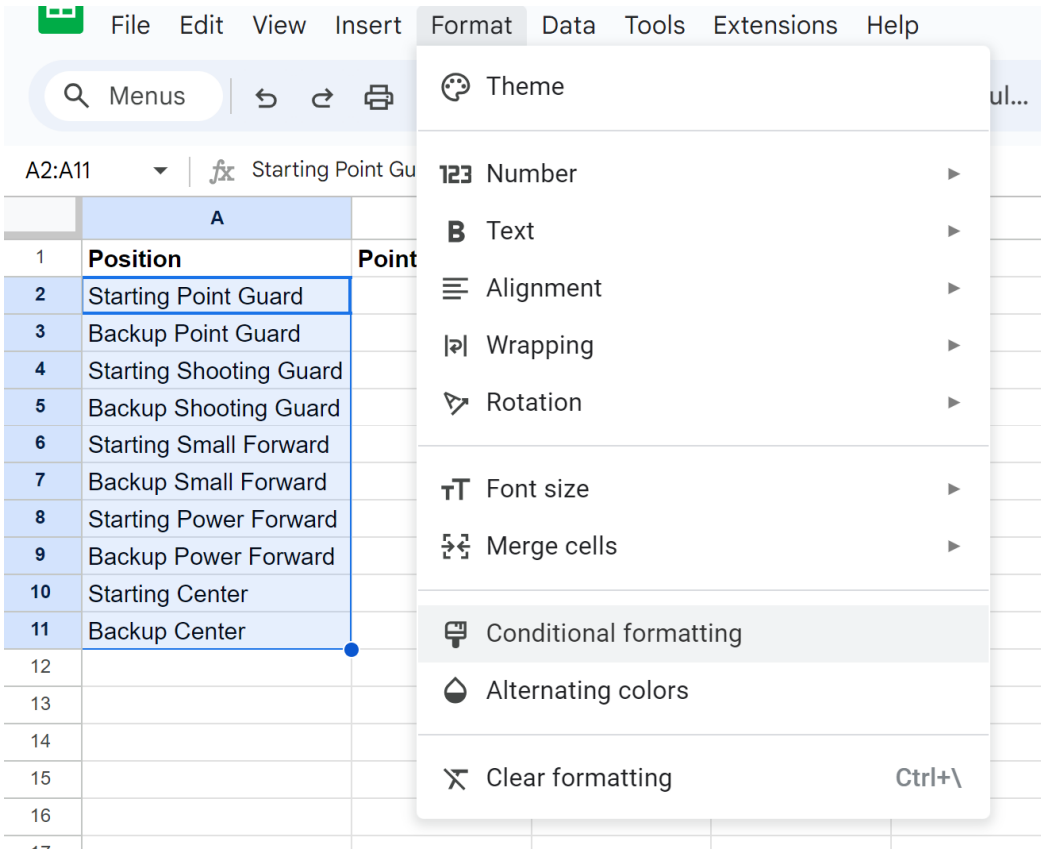
Starting
Forward
Center

This implementation ensures that regardless of how many positions need tracking, the solution remains scalable and efficient, relying on a single, comprehensive conditional rule rather than managing three or more separate rules that could conflict or become difficult to maintain.

## Step-by-Step Implementation of the Conditional Rule

Implementing the rule begins by clearly defining the scope of the formatting. First, select the precise range of cells that you wish to affect. In this scenario, since we are targeting the Position column, we must highlight the data range from the first data entry down to the last, specifically **A2:A11**. After defining the range, navigate to the main menu at the top of the Google Sheets interface, click the **Format** tab, and then select **Conditional formatting** from the resulting

dropdown menu.

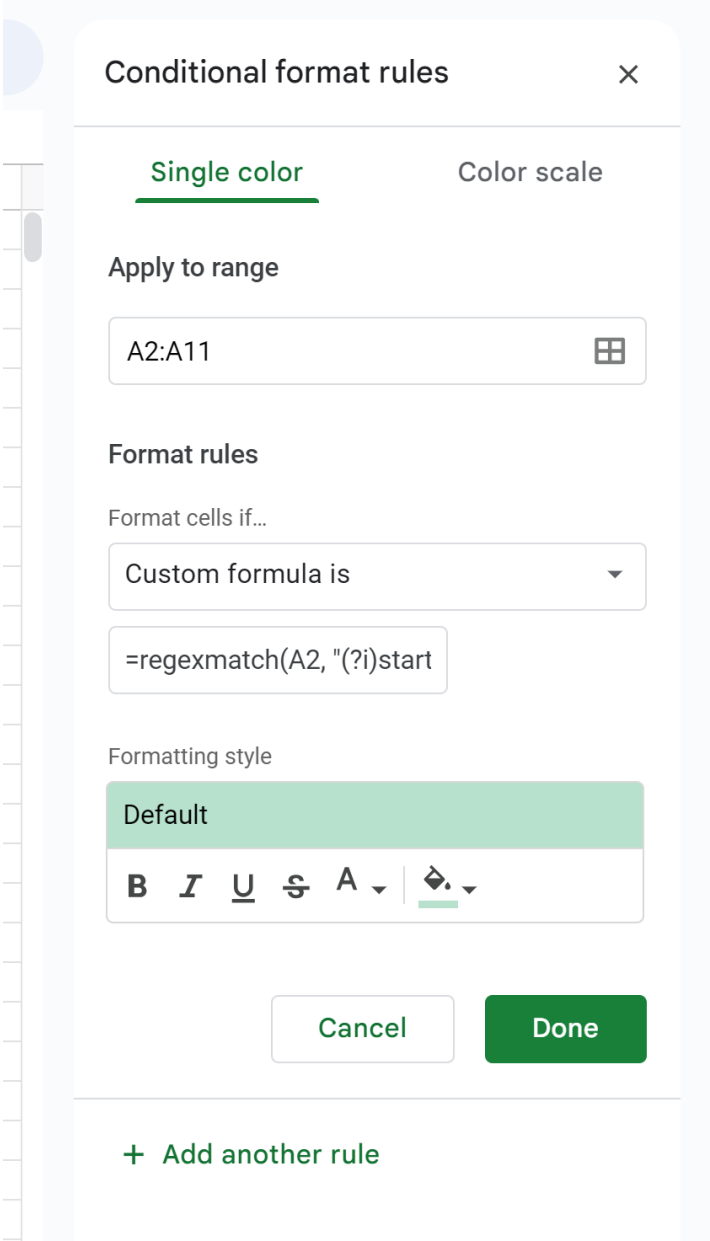



This action will open the dedicated **Conditional format rules** panel on the right side of your screen. Within this panel, you need to change the primary rule type. Click the **Format cells if** dropdown menu, scroll down the comprehensive list of options, and definitively choose the **Custom formula is** option. This critical selection signals to Google Sheets that the rule will be determined solely by the Boolean output of a complex formula rather than a standard, predefined condition.

Once the custom formula input field is active, input the precise formula designed to check for the multiple text values simultaneously. Remember that the formula must reference the first cell in the selected range (A2) without absolute referencing (no dollar signs). The formula required for this specific task is as follows:

**=regexmatch(A2, "(?i)starting|forward|center")**

After successfully entering the formula, you must choose the desired formatting style, such as a light green background color or a bold text style, before finally clicking **Done** to save and apply the rule instantly.




## Understanding the REGEXMATCH Formula

The core success of this advanced conditional formatting solution lies entirely in the **REGEXMATCH** function. This is a highly specialized tool in Google Sheets designed specifically for pattern matching using regular expressions (regex). When deployed within conditional formatting, this function checks if a specified string--the cell content--contains a match for the provided regular expression pattern.

Let's conduct a detailed dissection of the formula we used: `=regexmatch(A2, "(?i)starting|forward|center")`. The first argument, `A2`, is the starting cell in the applied range. Google Sheets is intelligent enough to automatically adjust this reference (A3, A4, A5, and

so on) as the rule is evaluated down the entire column. Crucially, the second argument, `"(?i)starting|forward|center"`, is the regular expression pattern itself, which must always be enclosed in double quotes for proper interpretation.

The pattern relies on two essential elements for our multi-value check. First, the `(?i)` prefix is an inline flag that modifies the behavior of the expression, making the entire match **case-insensitive**. Without this critical prefix, the formula would only match "starting" and would fail to match inputs like "Starting" or "STARTING." Second, the vertical bar (`|`) functions as the logical OR operator within the syntax of regular expressions. It instructs the function to return TRUE if the cell contains "starting" OR "forward" OR "center." This singular, comprehensive pattern efficiently encapsulates all three required conditions, achieving in one formula what would typically require managing multiple separate conditional rules.

## Reviewing the Results and Customization Options

Upon successfully defining and applying the custom conditional formatting rule, the visual result is immediate and highly effective. Each of the cells within the **Position** column (A2:A11) that contains any of the specified text values--Starting, Forward, or Center--will automatically be highlighted according to the chosen styling. This instant visual feedback is essential for data auditing and rapid analysis.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Position** | **Points** | | |
| 2 | Starting Point Guard | 22 | | |
| 3 | Backup Point Guard | 19 | | |
| 4 | Starting Shooting Guard | 14 | | |
| 5 | Backup Shooting Guard | 12 | | |
| 6 | Starting Small Forward | 30 | | |
| 7 | Backup Small Forward | 22 | | |
| 8 | Starting Power Forward | 34 | | |
| 9 | Backup Power Forward | 12 | | |
| 10 | Starting Center | 15 | | |
| 11 | Backup Center | 10 | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |

It is important to remember that the visual aesthetic is entirely controlled by the user. While we opted for a light green fill in this tutorial for clear demonstration, the conditional formatting panel offers extensive customization capabilities. You have the freedom to choose any combination of background color, text color, font styling (such as bold or italic), or even border styles. This immense flexibility allows you to align the formatting precisely with your organization's established data visualization standards or personal reporting preferences.

Furthermore, this methodology is inherently flexible and is not strictly limited to exact matches. Because regular expressions look for the pattern *within* the string by default, if a cell contained the text "Backup Forward," the rule would still highlight it successfully because the substring "forward" is present. This behavior provides immense utility for matching partial or complex text entries, making the REGEX approach superior to simple "Text is exactly" rules.

## Troubleshooting and Advanced Text Matching Techniques

While **REGEXMATCH** is highly efficient, users sometimes encounter common errors during implementation. A frequent issue is failing to enclose the regular expression pattern in double quotes, which results in an immediate parse error. Another common oversight is forgetting the `(?i)` flag, leading to frustrating missed matches if the text casing varies throughout the dataset (e.g., matching "center" but failing on "Center"). Always double-check that the starting cell reference (A2 in our example) correctly reflects the top-left cell of the applied range and that the range itself is accurate.

For scenarios requiring more precise control, advanced regular expression techniques can be seamlessly integrated. For instance, if you strictly required an exact match--meaning the cell *must only* contain "Starting" and nothing else--you would introduce specific boundary anchors into your pattern: `="^starting$|^forward$|^center$"`. The caret (`^`) denotes the absolute start of the string, and the dollar sign (`$`) denotes the absolute end of the string. This implementation enforces strict, whole-cell matching, preventing the rule from highlighting cells that might contain additional descriptive text like "Starting Rookie" or "Forward Position Backup."

Mastering these nuances of pattern matching and leveraging the custom formula feature significantly enhances your ability to manage large and complex datasets efficiently within Google Sheets, transforming raw, dense data into actionable, visually distinct information for rapid decision-making.

## Additional Resources for Google Sheets Mastery

The following tutorials explain how to perform other common tasks and advanced data manipulation techniques in Google Sheets: