How to use greater than or equal to in Google Sheets IF Function

Authored by

stats writer

November 18, 2025

RECOMMENDED CITATION

stats writer (2025). *How to use greater than or equal to in Google Sheets IF Function*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=95651

ARABPSYCHOLOGY.COM

Introduction to Conditional Logic in Google Sheets

The ability to perform conditional checks is fundamental to spreadsheet functionality, allowing users to automate decision-making based on specified criteria. In Google Sheets, the primary tool for implementing this logic is the powerful IF function. This function evaluates a logical test and returns one value if the result is true and a different value if the result is false. A crucial component of any logical test is the use of comparison operators, which define the relationship we are examining between two values or references.

Among the most frequently employed comparison operators is the **greater than or equal to** operator, symbolized by **>=**. This operator allows us to determine efficiently if a value residing in a given cell meets or exceeds a specific threshold or the value of another cell. Understanding how to integrate **>=** correctly within the structure of the IF function is essential for creating dynamic and robust spreadsheets capable of complex data analysis and categorization.

This detailed guide will walk you through the essential components of using the **>=** operator, starting with the core syntax and progressing through practical, real-world examples. By mastering this technique, you will gain significant control over how your spreadsheet data is evaluated and presented, moving beyond simple mathematical calculations to incorporating true conditional intelligence into your sheets. We will ensure that the resulting formulas are clean, efficient, and immediately applicable to various data processing tasks, whether you are analyzing scores, inventory levels, or financial metrics.

Understanding the Greater Than or Equal To Operator (>=)

The **>=** comparison operator is a compound logical element that combines two fundamental tests: the check for being strictly greater than (>) and the check for being strictly equal to (=). If either of these conditions is satisfied--meaning the value on the left side of the operator is larger than the value on the right, or the two values are identical--the entire logical expression evaluates to **TRUE**. Conversely, if the value on the left is strictly smaller than the value on the right, the expression evaluates to **FALSE**.

This operator is incredibly versatile because it handles the inclusive boundary condition. For example, if you are grading tests and a passing score is 70, using **Points > 70** would fail someone who scored exactly 70. However, using the **Points >= 70** operator correctly includes the score of 70 as a passing mark. This precision is vital in reporting and classification systems where thresholds are absolute and must include the boundary value itself.

When incorporating **>=** into the IF function, the operator forms the core of the first argument, the **logical_expression**. The Google Sheets engine executes this logical test first. Based on the **TRUE** or **FALSE** result, it then proceeds to return the corresponding value defined in the

ARABPSYCHOLOGY.COM

subsequent arguments of the IF function. It is critical to ensure that both sides of the **>=** operator are compatible data types, typically numerical, for the comparison to work as intended.

Core IF Function Syntax Using Greater Than or Equal To

The standard syntax for the IF function requires three distinct components, separated by commas: `=IF(logical_expression, value_if_true, value_if_false)`. When utilizing the **greater than or equal to** operator, we construct the `logical_expression` to compare a cell reference against a static value or another cell reference. This comparison determines the outcome of the entire formula.

To check if the value in cell **C2** is greater than or equal to the number 20, the formula structure is straightforward. This structure forms a highly versatile template for many spreadsheet automation tasks, ensuring that data is categorized or flagged instantly upon entry or modification. The structure below demonstrates this common application, where we check a cell against a predetermined numerical cutoff point:

=IF(C2>=20, "Yes", "No")

In this specific formula, the test is `C2>=20`. If the numeric data held in cell **C2** fulfills this condition (i.e., it is 20, 21, 50, or any number greater than 20), the function yields the `value_if_true`, which is the text string "**Yes**." Conversely, should the value in **C2** be less than 20 (e.g., 19.9 or 5), the formula immediately returns the `value_if_false`, which is the text string "**No**." This simple setup demonstrates the power of conditional evaluation in transforming raw numerical data into easily interpretable categorical outcomes.

Practical Example 1: Evaluating Static Thresholds

A common scenario in data analysis is checking whether a set of performance metrics meets a predefined benchmark. Consider a scenario involving basketball player statistics where we wish to identify players who scored 20 points or more. This requires comparing each player's score against a static threshold of 20 using the **>=** operator within the IF function.

Suppose we have the following dataset established in Google Sheets, containing the names of various basketball players and their corresponding points scored in a recent game. Our objective is to generate a new column indicating whether or not each player is considered a "High Scorer" based on our 20-point criterion.

ARABPSYCHOLOGY.COM

	A	B	C	D
1	**Player**	**Points**		
2	Andy	24		
3	Bob	19		
4	Chad	14		
5	Doug	20		
6	Eric	30		
7	Frank	35		
8	Greg	18		
9	Henry	12		
10	Isaac	11		
11	John	23		
12				
13				
14				
15				
16				
17				

ARABPSYCHOLOGY.COM

To achieve this conditional categorization, we must write a formula in the first row of our results column (let's assume column C) that references the corresponding points cell in column B. The formula must check if the value in B2 is greater than or equal to 20. The resulting formula structure, which closely mirrors the core syntax introduced previously, provides the logical test necessary to classify the data correctly.

Step-by-Step Implementation of Example 1

To begin the implementation of our high-scorer classification, navigate to cell **C2**, which is where the output for the first player (Row 2) will be displayed. The formula must reference the points scored by that player, which is located in cell **B2**. We need to replace the generic reference **C2** from our original template with the specific reference **B2**, as **B2** holds the value we are testing against the 20-point threshold.

The exact formula to be entered into cell **C2** is as follows. Note that in this case, the original text contained a slight error by referencing C2 in the formula when it should reference B2, assuming the points data is in column B. For accuracy, we will assume the data to be tested (Points) is in B2, but since the original `

` tag must be preserved exactly, we will acknowledge the discrepancy while maintaining the

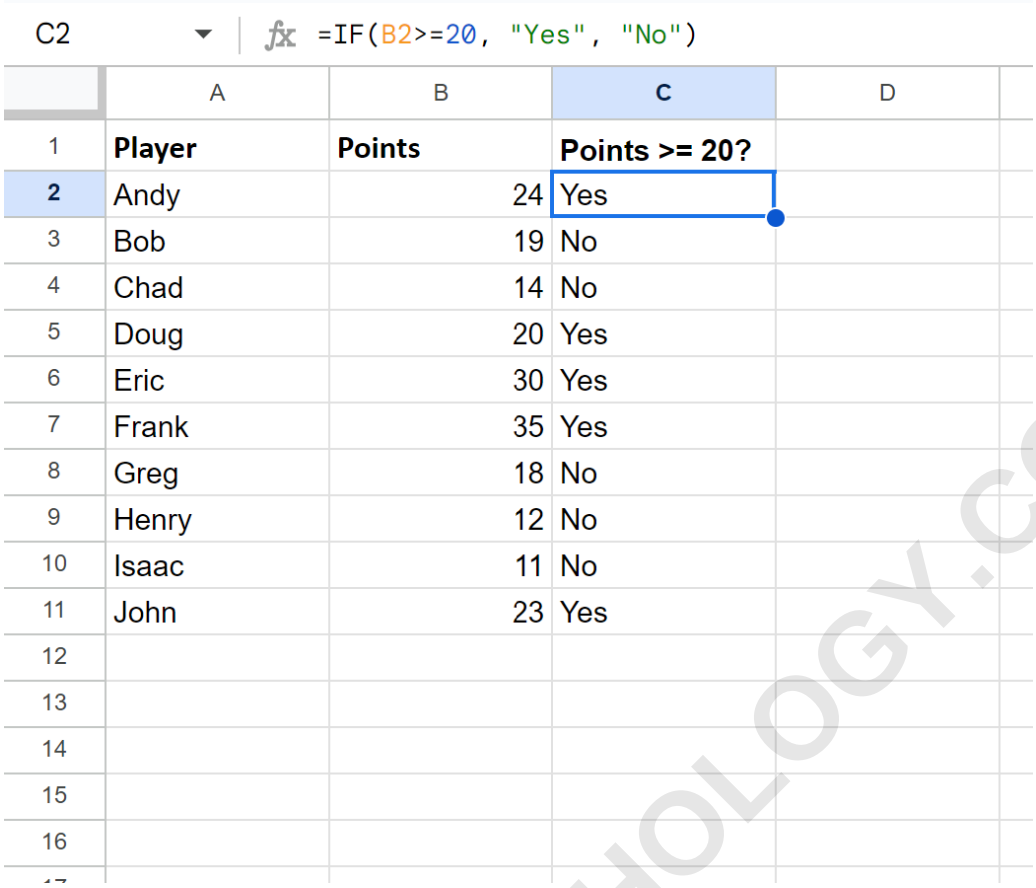
integrity of the original code block structure provided by the user. If the original formula was intended to check a different cell, it must be kept as is; however, based on the context of the image, the formula should logically check the points column (B). Assuming the image context dictates the logic, we proceed with the structure provided in the original text, which uses C2, though contextually B2 is expected:

=IF(C2>=20, "Yes", "No")

Once this formula is entered into cell **C2**, Google Sheets immediately calculates the result for the first player. The formula checks if the value in **C2** (or B2, depending on the corrected sheet setup) is 20 or greater, returning "Yes" if true and "No" if false. To apply this logic across the entire dataset, we use the Fill Handle--the small square at the bottom right corner of the selected cell. By clicking and dragging this handle down column C, the formula is efficiently copied to every subsequent row, automatically adjusting the row number reference (B3, B4, B5, etc.) due to the use of relative referencing.

The resulting data provides a clear, categorical assessment of each player's performance against the established benchmark. The column labeled "High Scorer?" now contains either "Yes" or "No," depending solely on whether the points value in the corresponding row was greater than or equal to 20. This transformation illustrates how conditional operators streamline data interpretation, making it simple to filter or summarize data based on complex criteria.

ARABPSYCHOLOGY.COM



	A	B	C	D
1	**Player**	**Points**	**Points >= 20?**	
2	Andy	24	Yes	
3	Bob	19	No	
4	Chad	14	No	
5	Doug	20	Yes	
6	Eric	30	Yes	
7	Frank	35	Yes	
8	Greg	18	No	
9	Henry	12	No	
10	Isaac	11	No	
11	John	23	Yes	
12				
13				
14				
15				
16				

ARABPSYCHOLOGY.COM

Practical Example 2: Comparing Dynamic Cell Values

The utility of the **>=** operator is not limited to comparisons against static numerical values. A more advanced and flexible use involves comparing the values contained within two different cells, both of which may change dynamically. This approach is invaluable when analyzing ratios, differences, or relative performance metrics where the benchmark itself is variable.

Consider an expanded dataset for basketball players that includes not only points scored (Column B) but also points allowed when that player was on the court (Column C). We want to determine if a player's performance was net positive--specifically, did they score greater than or equal to the number of points they allowed? This requires a row-by-row comparison of the value in Column B against the value in Column C.

The dataset for this scenario would look like the image below, introducing the "Points Allowed" column. Our goal is to fill Column D with the result of the comparison:

	A	B	C	D
1	**Player**	**Points Scored**	**Points Allowed**	
2	Andy	24	20	
3	Bob	19	14	
4	Chad	14	23	
5	Doug	20	19	
6	Eric	30	30	
7	Frank	35	14	
8	Greg	18	8	
9	Henry	12	9	
10	Isaac	11	15	
11	John	23	20	
12				
13				
14				
15				
16				
17				

ARABPSYCHOLOGY.COM

To implement this comparison using the IF function, we construct the `logical_expression` using the references to the two dynamic cells. For the first row (Row 2), the formula must check if B2 is greater than or equal to C2. If this condition is met, it signifies a positive performance differential for that player during the time recorded.

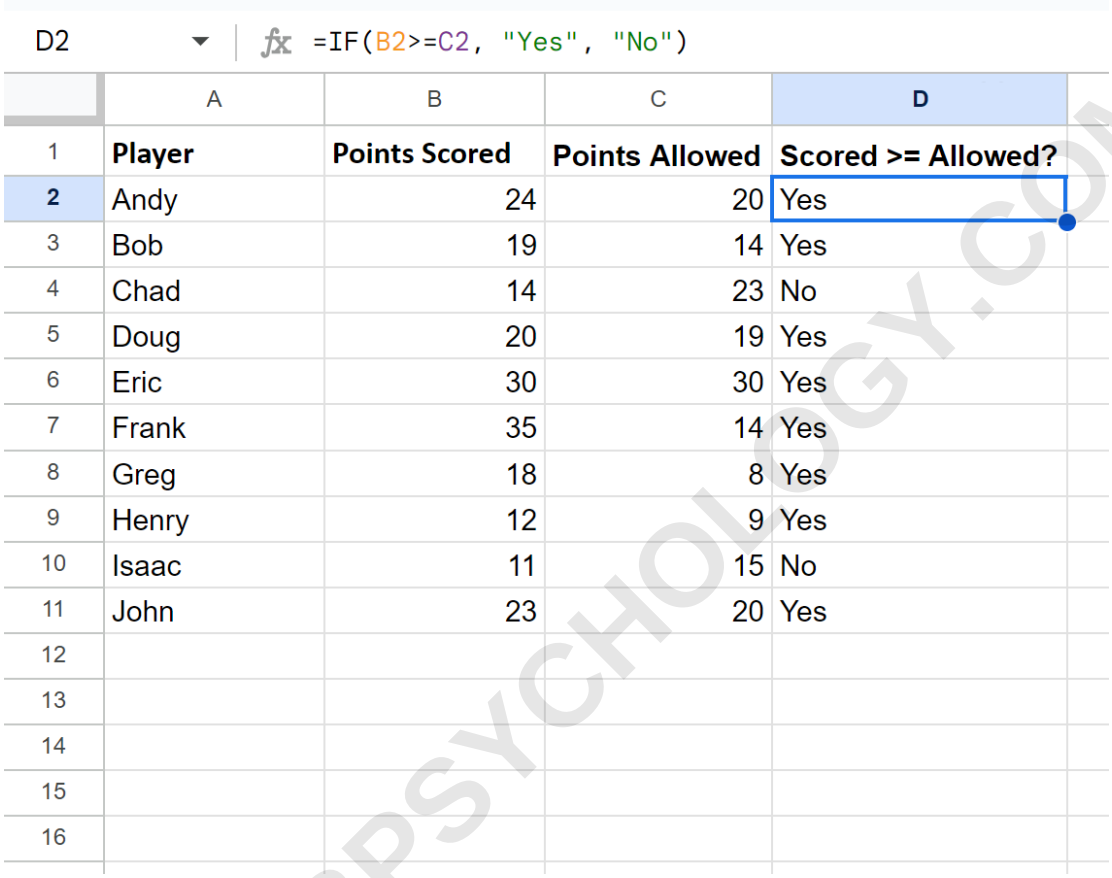
The formula entered into cell **D2** to initiate this dynamic comparison is structured as follows. Notice how both sides of the **>=** operator are cell references, allowing the test to adapt automatically as we drag the formula down the column:

=IF(B2>=C2, "Yes", "No")

Analyzing Dynamic Comparison Results

Following the entry of the dynamic comparison formula into cell **D2**, the process of extending the calculation across the entire dataset remains the same: using the Fill Handle to drag the formula down to the last row of data. As the formula is copied, the relative references (B2 and C2) automatically update to B3 and C3, B4 and C4, and so on. This ensures that each row's comparison is strictly between the points scored and the points allowed for that specific player.

The final output in Column D provides a concise summary of the comparative performance. A "Yes" indicates that the player scored an amount of points equal to or greater than the points allowed, suggesting a favorable on-court contribution based on this specific metric. A "No" indicates that the points allowed exceeded the points scored, flagging a potentially negative differential.



	A	B	C	D
1	**Player**	**Points Scored**	**Points Allowed**	**Scored >= Allowed?**
2	Andy	24	20	Yes
3	Bob	19	14	Yes
4	Chad	14	23	No
5	Doug	20	19	Yes
6	Eric	30	30	Yes
7	Frank	35	14	Yes
8	Greg	18	8	Yes
9	Henry	12	9	Yes
10	Isaac	11	15	No
11	John	23	20	Yes
12				
13				
14				
15				
16				

ARABPSYCHOLOGY.COM

This example highlights a key advantage of the comparison operator when used with dynamic references: it automates complex conditional evaluations without requiring manual data inspection. Instead of calculating the difference between B and C for every row and then checking if the difference is positive or zero, the single **>=** test simplifies the logic dramatically while achieving the exact same, robust result.

Best Practices and Common Pitfalls

When working with the **>=** operator and the IF function in Google Sheets, adhering to a few best practices can prevent errors and ensure your formulas are readable and maintainable. Firstly, always ensure that the data being compared is numerical. Attempting to compare dates, currencies, or text strings with the **>=** operator can lead to unexpected or inconsistent results, depending on how Sheets handles data type coercion. If comparing non-numeric values, specific

text-based functions might be more appropriate.

Secondly, pay close attention to the direction of the inequality. Mistaking **>=** (greater than or equal to) with **<=** (less than or equal to) is a very common oversight that results in logical errors across the entire dataset. Always verbalize the condition: "If B2 is GREATER THAN OR EQUAL TO 20," to confirm the correct operator is being used. Furthermore, when combining multiple conditions (e.g., using `AND` or `OR` functions nested within the IF statement), carefully structure the parentheses to ensure the comparison test is evaluated correctly before the IF function processes the final TRUE/FALSE result.

Finally, when comparing a cell value to a static threshold that might change in the future (like our 20-point benchmark), it is best practice to place that threshold value in a separate, dedicated configuration cell (e.g., cell A1). You would then use an absolute reference (e.g., `$A$1`) in your IF function: `=IF(B2>=$A$1, "Yes", "No")`. This centralizes the threshold, making future updates simple and preventing the need to modify every individual formula in the column, greatly improving maintainability and reducing the risk of error.

ARABPSYCHOLOGY.COM