Excel Formula: If Between Two Numbers then Return Value

Authored by

stats writer

November 17, 2025

Achieving Conditional Value Retrieval in Microsoft Excel

In data analysis and financial modeling, a common requirement is to conditionally extract data points based on whether they satisfy specific criteria. One of the most frequent conditional checks involves determining if a numerical value falls within a defined, inclusive range. Microsoft Excel, the industry standard for spreadsheet management, provides powerful logical functions that enable users to execute this precise test efficiently. By nesting the IF function with the AND function, we can construct a robust formula designed to return the original value only if it meets both the lower and upper boundary conditions simultaneously.

The fundamental structure for this conditional check is crucial for isolating specific data entries that adhere to certain constraints, such as identifying scores within a passing band or sales figures within a target quota. Utilizing this combined logical approach ensures that the output is precise, enhancing data cleanliness and facilitating deeper analysis. This technique moves beyond simple filtering by providing an automated mechanism to extract values directly into a separate column, allowing for subsequent calculations or visualizations involving only the qualifying data.

The standard syntax used to implement this specific range check--where the value must be greater than or equal to the lower bound and less than or equal to the upper bound--is presented below. This formula is designed to check the value contained within a specified cell, compare it against the established limits, and either return the value itself or an empty string, signifying non-compliance with the criteria.

=IF(AND(B2>=20, B2<=30), B2, "")

This powerful construction specifically examines the content of cell **B2** to determine if it falls inclusively between the numerical range of **20** and **30**. The nested AND function establishes the mandatory requirements: the value must be 20 or greater, and concurrently, 30 or less. Should both logical conditions evaluate to TRUE, the outer IF function executes its true value argument, which in this case is the original value stored in **B2**. Conversely, if either condition is FALSE, the formula returns a blank string ("").

Dissecting the Logic: How IF and AND Functions Work Together

To fully appreciate the efficiency of this solution, it is essential to understand the roles played by the two primary logical functions: IF and AND. The IF function serves as the primary decision-maker, governing the output based on a single logical test. Its structure is straightforward: `IF(logical_test, value_if_true, value_if_false)`. However, when checking for a range, we require two logical tests to be true simultaneously--a requirement that exceeds the capability of a standalone IF function.

This is where the AND function becomes indispensable. The AND function takes multiple logical arguments and returns TRUE only if all arguments evaluate to TRUE. If even one argument is FALSE, the AND function returns FALSE. By embedding the entire AND statement--which encompasses both the lower boundary test (`B2>=20`) and the upper boundary test (`B2<=30`)--as the `logical_test` argument within the IF function, we condense two separate conditions into a single, executable TRUE/FALSE result.

Consider the components of the formula `=IF(AND(B2>=20, B2<=30), B2, "")`. The first argument within the AND function, `B2>=20`, checks the lower boundary, ensuring the value is not below 20. The second argument, `B2<=30`, checks the upper boundary, ensuring the value does not exceed 30. Only when the value in cell B2 is 20, 30, or any number in between, will the AND function return TRUE. This TRUE signal then tells the IF function to proceed to its second argument (`B2`), effectively returning the number itself. If the AND function returns FALSE, the IF function defaults to the third argument (`""`), resulting in a blank display. This nesting is a fundamental paradigm for conducting complex, multi-criteria evaluations in Microsoft Excel.

Practical Application: Filtering Data Based on Performance Metrics

To illustrate the practical utility of this conditional extraction formula, let us apply it to a typical scenario involving performance dataset analysis. Imagine we are managing records for a basketball team and need to isolate the scores of players who achieved a specific target range of points during a game. This selection criterion is vital for analyzing players who performed moderately well but did not hit outlier high or low scores.

Suppose we are working with the following dataset in Microsoft Excel, which contains the names of various players in Column A and the corresponding points they scored in Column B. Our analytical goal is to filter and display only those point totals that fall inclusively between 20 and 30 points, leaving the rest of the result cells empty for clarity. This focused approach allows us to quickly view the subset of data that meets the specified performance benchmark.

The original data structure appears as depicted in the image below, where Column B holds the crucial numerical information we intend to test against our predefined range. This step-by-step application ensures that the implementation of the complex logical formula is clear and easily replicable across different analytical tasks within the spreadsheet environment.

	A	B	C	D	E
1	**Player**	**Points**			
2	Andy	28			
3	Bob	31			
4	Chad	25			
5	Doug	19			
6	Eric	14			
7	Frank	17			
8	Greg	22			
9	Henry	28			
10	Isaac	34			
11	John	20			
12	Kendall	23			
13	Luke	46			
14					
15					
16					

Implementing the Formula for Conditional Extraction

To execute the required range check, we will enter the logical formula into the first row of a new column, typically Column C, which will serve as our output field. This column will either display the qualifying point total from Column B or remain blank if the score is outside the 20-30 range. We begin by entering the formula into cell **C2**, targeting the first data point in cell B2.

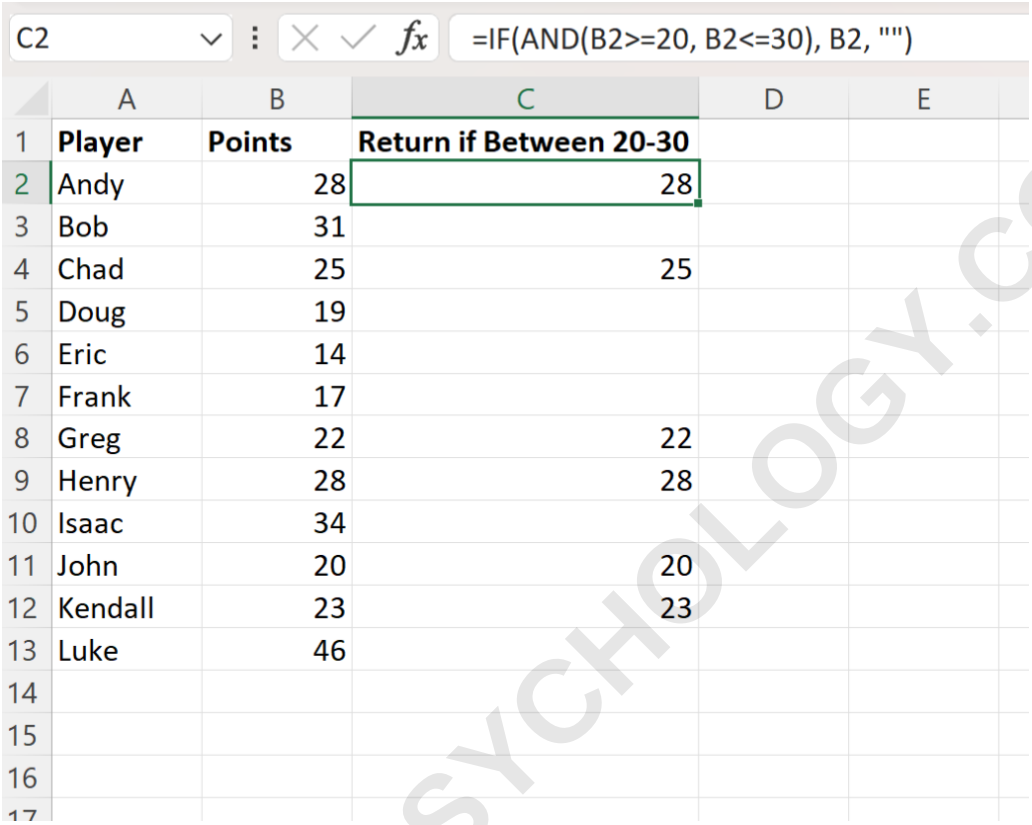
The formula used to check if the value in the **Points** column meets the specified criteria (20 ≤ Points ≤ 30) is constructed as follows. Notice the inclusion of the equality operator (>= and <=), which confirms that the boundary numbers 20 and 30 themselves are included in the qualifying range. If exclusive boundaries were desired (i.e., strictly between 20 and 30), the equality signs would be removed.

=IF(AND(B2>=20, B2<=30), B2, "")

Once this formula is correctly entered into cell **C2**, the subsequent step involves applying this identical logic dynamically to the rest of the dataset. This is efficiently achieved by utilizing Excel's autofill feature. We simply click on the lower-right corner of cell **C2** (the fill handle) and drag the formula down to cover all corresponding rows in Column C. Excel automatically adjusts the relative cell references (e.g., changing `B2` to `B3`, `B4`, and so on), ensuring that each row is evaluated

against the constant range criteria.

The visual result after performing the drag-and-fill operation clearly delineates the values that satisfy the condition from those that do not. This transformation turns a simple list of numbers into a conditionally filtered output, immediately highlighting the target subset of data points.



	A	B	C	D	E
1	**Player**	**Points**	**Return if Between 20-30**		
2	Andy	28	28		
3	Bob	31			
4	Chad	25	25		
5	Doug	19			
6	Eric	14			
7	Frank	17			
8	Greg	22	22		
9	Henry	28	28		
10	Isaac	34			
11	John	20	20		
12	Kendall	23	23		
13	Luke	46			
14					
15					
16					

Interpreting the Output and Understanding Blank Returns

Analyzing the resulting Column C provides immediate confirmation of the formula's successful execution. For every row where the value in Column B (Points) is strictly between 20 and 30, inclusive of the boundaries, the corresponding cell in Column C displays that exact value. This represents the 'value_if_true' argument (`B2`) being activated by the successful logical test performed by the AND function.

Conversely, if the score in Column B is 15 (less than 20) or 35 (greater than 30), the AND function evaluates to FALSE because at least one of the two logical requirements is not met. Consequently, the outer IF function executes its 'value_if_false' argument, which we defined as an empty string (`""`). This mechanism efficiently suppresses the non-qualifying data points, making the resulting column a filtered view of the original numerical dataset without altering the source data itself.

The decision to return a blank (`""`) instead of a zero (`0`) or an error message is often preferred in

data presentation, as blanks are visually cleaner and do not interfere with subsequent mathematical operations that might incorrectly include zeros. This method of conditional extraction is particularly useful when preparing data for reporting, as it provides a clean, concise subset of qualifying figures ready for charts or further summarization.

Modifying Output: Returning Text Classifications (Yes/No)

While the previous example focused on extracting the numerical value itself, the flexibility of the IF function allows for immense customization regarding the output. Instead of returning the raw number, analysts often prefer to return a categorical classification, such as "Yes" or "No," to clearly indicate whether the value met the criteria. This is particularly useful in audit trails or validation columns where a clear binary result is required.

To switch the output from numerical extraction to categorical classification, we simply modify the second and third arguments of the main IF formula. The logical test--the embedded AND function--remains identical, as the criteria for evaluating the range have not changed. Only the response triggered by the TRUE or FALSE outcome is adjusted.

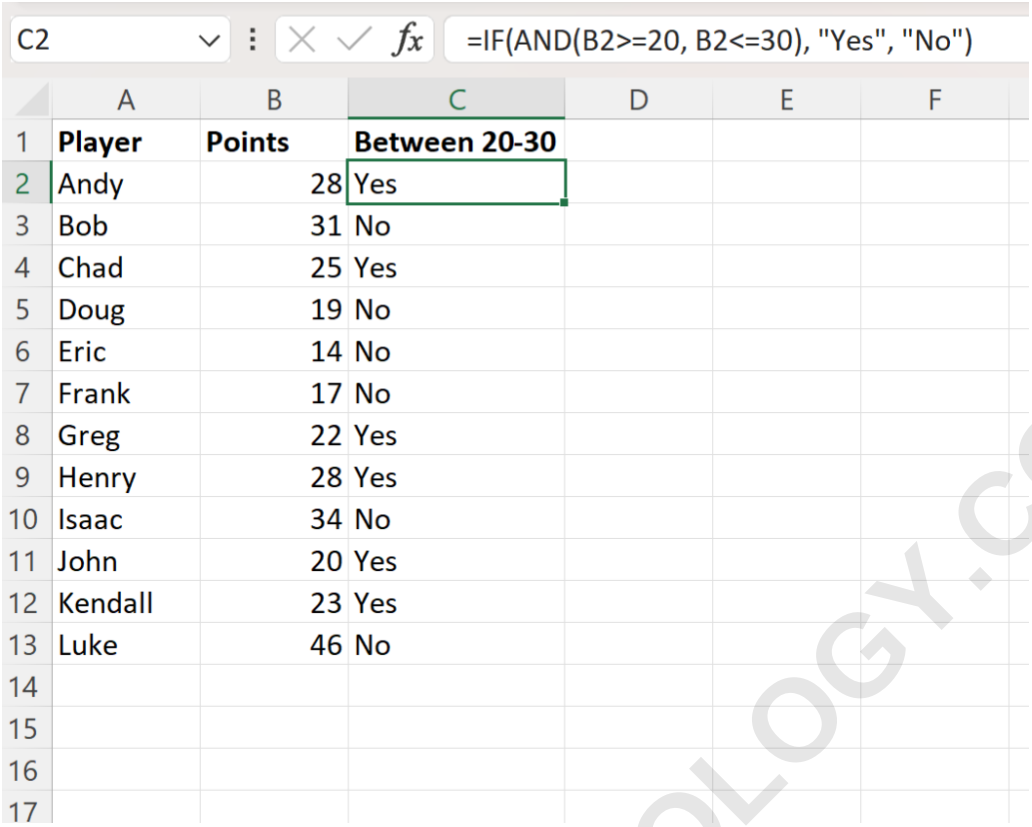
For instance, to return "Yes" if the score is between 20 and 30 (inclusive) and "No" otherwise, the structure of the formula is adapted as shown below. The value returned if TRUE is now the string `"Yes"`, and the value returned if FALSE is the string `"No"`. Note that text strings must always be enclosed in double quotes within Microsoft Excel formulas.

=IF(AND(B2>=20, B2<=30), "Yes", "No")

Visualizing Categorical Range Checks

Upon applying this modified formula to the same basketball points dataset, the result in Column C shifts entirely to text-based categorical indicators. This visual representation is sometimes more intuitive for non-technical users, offering an immediate assessment of compliance against the 20-to-30 point range.

The following screenshot demonstrates the application of the Yes/No formula variant, highlighting how the corresponding value in Column C now clearly communicates the outcome of the range check performed on the numerical data in Column B. This provides a clear, Boolean-like result for every entry.



	A	B	C	D	E	F
1	**Player**	**Points**	**Between 20-30**			
2	Andy	28	Yes			
3	Bob	31	No			
4	Chad	25	Yes			
5	Doug	19	No			
6	Eric	14	No			
7	Frank	17	No			
8	Greg	22	Yes			
9	Henry	28	Yes			
10	Isaac	34	No			
11	John	20	Yes			
12	Kendall	23	Yes			
13	Luke	46	No			
14						
15						
16						
17						

As is evident from the output, Excel has successfully translated the complex numerical range verification into simple, explicit text labels. Row entries like the score of 25 (which is between 20 and 30) now display "Yes," while entries like 15 or 35 display "No." This categorical output is a robust method for creating validation flags or status indicators within large spreadsheets, significantly aiding in auditing and reporting processes.

Advanced Considerations: Inclusive vs. Exclusive Boundaries

When implementing range checks, it is critical to select the appropriate comparison operators, as this defines whether the boundaries themselves are included in the result. The examples provided thus far utilize **inclusive boundaries**, meaning that the lower limit (20) and the upper limit (30) are both considered valid results. This is achieved through the use of the 'greater than or equal to' (`>=`) and 'less than or equal to' (`<=`) operators within the AND function.

If the requirement shifts to an **exclusive range**--where the value must be strictly greater than the lower limit and strictly less than the upper limit (e.g., all numbers between 20 and 30, but excluding 20 and 30 themselves)--the formula must be adjusted accordingly. The operators would change from `>=` and `<=` to simply `>` and `<`. For instance, to check if B2 is strictly between 20 and 30, the logical test would become: `AND(B2 > 20, B2 < 30)`. This distinction is subtle but paramount for accurate data validation, especially in highly precise analytical contexts.

Furthermore, this nested IF(AND) structure can be expanded to check multiple ranges or introduce additional criteria using other logical operators like the OR function. For example, if a value needs to be between 20 and 30 OR between 50 and 60, the structure would require embedding multiple AND functions within a single OR function: `IF(OR(AND(B2>=20, B2<=30), AND(B2>=50, B2<=60)), B2, "")`. Mastering the combination of these logical building blocks is key to performing advanced conditional logic within Microsoft Excel.

Summary of Conditional Range Evaluation

The ability to conditionally evaluate whether a numerical entry falls between two predefined boundaries is a cornerstone of effective data manipulation in Microsoft Excel. By expertly combining the IF function and the AND function, users gain a precise tool for filtering, extracting, and classifying data points based on complex, simultaneous criteria. Whether the goal is to return the value itself for further calculation or to provide a clear categorical indicator like "Yes" or "No," the logical framework remains highly adaptable.

We have demonstrated that the core formula relies on two critical conditions being met for success: the value must meet or exceed the lower bound, and it must be less than or equal to the upper bound. This technique is invaluable across diverse applications, ranging from quality control checks and financial threshold identification to scientific data filtering. Mastery of this nesting concept significantly enhances a user's ability to conduct sophisticated analyses and automate data cleansing processes within large datasets.

Remember that clarity in defining the range--specifically whether boundaries are inclusive or exclusive--is paramount, as a simple change in the comparison operator (> vs. >=) fundamentally alters the output. Consistent application of this powerful logical formula ensures data integrity and analytical precision.