Learn How to Check if a Number is Between Two Values Using Excel’s IF and AND Functions

Authored by

Mohammed looti

November 10, 2025

Mastering Conditional Range Checks in Excel

The ability to perform conditional checks based on numerical ranges is fundamental to advanced **data processing** within spreadsheet applications. When analyzing large datasets, users frequently encounter the need to extract or flag values that fall within a precise upper and lower boundary. Fortunately, Excel provides a straightforward yet powerful method for achieving this using the nested combination of the **IF function** and the **AND function**, allowing for sophisticated logical evaluations. This structure is essential because it ensures that two separate conditions--the value being greater than or equal to the lower limit, and simultaneously less than or equal to the upper limit--must be true for the desired outcome to be generated.

This logical combination forms the backbone of efficient **data filtering** and extraction when dealing with quantitative information. The primary objective of this technique is to return a specific value, usually the original data point itself, only if it fully satisfies the predefined range criteria. Should the value lie outside this range, the IF function defaults to an alternative result, such as a blank cell or a customized text string. By strategically leveraging this IF(AND) structure, analysts can fully automate the process of isolating relevant data points from expansive spreadsheets, eliminating the need for tedious manual checking.

To demonstrate this principle, we utilize a core formula designed for returning a value in Excel, provided it falls inclusively between two specified numbers. This specific formula checks the contents of cell **B2** against the inclusive boundaries of 20 and 30.

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

Deconstructing the IF(AND) Formula Structure

Understanding the syntax and the precise role of each component is crucial for customizing this logic for diverse datasets and requirements. The IF function operates based on three primary arguments: the logical test, the value if true, and the value if false. In our specific implementation, the logical test itself is composed of the nested AND function, which is necessary because checking a range inherently requires the evaluation of multiple, simultaneous conditions.

The internal structure, `AND(Condition1, Condition2)`, evaluates the input and only returns **TRUE** if both Condition 1 (`B2>=20`) and Condition 2 (`B2<=30`) are met. This is a fundamental concept of Boolean logic. For example, if the value in **B2** is 25, both conditions are true, and the AND function returns TRUE. Conversely, if **B2** is 15, the first condition is false, causing the entire AND function to return FALSE. This definitive TRUE/FALSE output is then seamlessly passed back as the logical test for the outer IF function.

If the nested AND function evaluates to TRUE, the IF function executes its second argument, which in the initial example is simply `B2`--meaning it efficiently returns the original value from the cell. Conversely, if the AND function returns FALSE, the IF function executes its third argument, which is defined as `""`, signifying an empty string or a blank result. This structure allows us to efficiently filter and extract data, ensuring only values that satisfy the specific range criteria are displayed in the results column, providing highly targeted data extraction.

Practical Application: Filtering Dataset Values

To fully illustrate the robust utility of this formula, we will examine a practical scenario involving performance metrics. Imagine we are tasked with analyzing a dataset containing points scored by various basketball players. Our objective is to isolate only those players whose scores fall within a specific, desirable range--for instance, between 20 and 30 points inclusive. This precise filtering process is vital for identifying a mid-tier performance group, effectively excluding outliers who scored either too low or exceptionally high.

The following image represents the raw dataset we are working with. Column A lists the Player names, and Column B contains the corresponding Points scored. Our goal is to leverage Column C to display only the points that satisfy our target criteria (scores between 20 and 30).

	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					

We initiate the filtering process by applying the nested formula in the first available cell of the

results column, which is **C2**. The formula specifically targets the corresponding numerical value in Column B. It is essential to ensure that we use the correct comparison operators (`>=` for greater than or equal to, and `<=` for less than or equal to) to include the boundary numbers (20 and 30) within our acceptable range. This inclusion creates an inclusive range, which is standard practice in many statistical analyses unless an exclusive range is explicitly required.

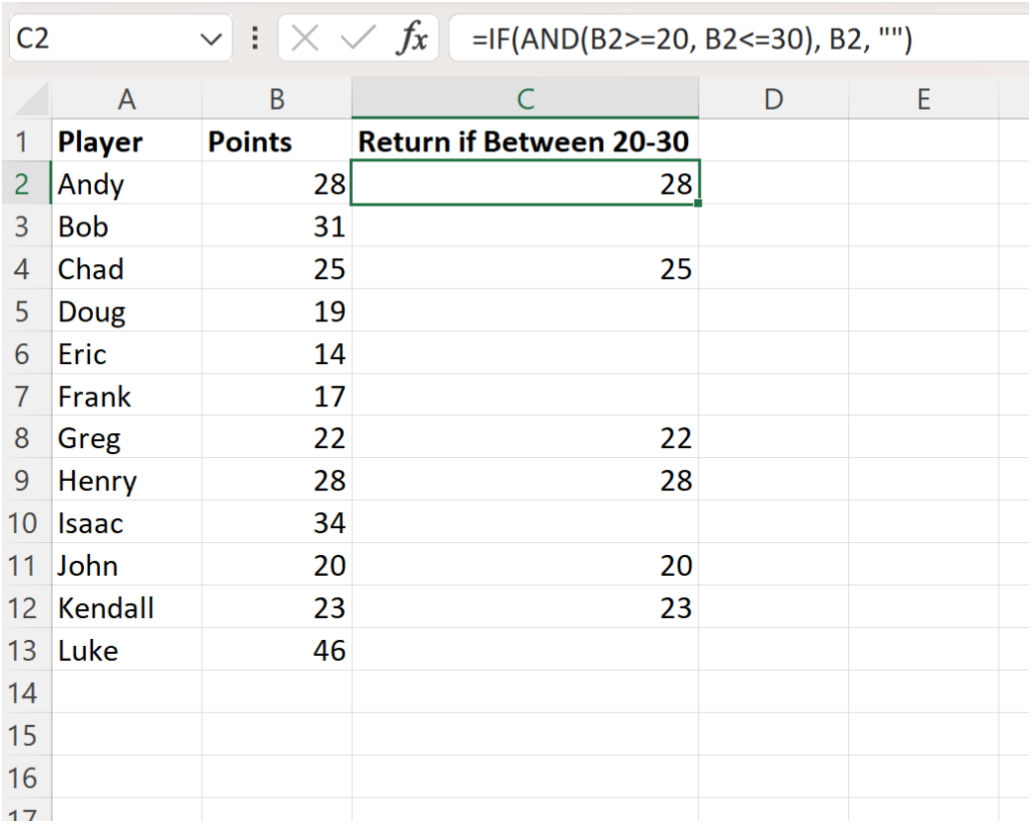
Implementing the Formula to Return the Value

The specific formula we input into cell **C2** is explicitly designed to check the value in **B2** and, if the criteria are met, simply echo that value back into the results column. This method is exceptionally useful when the objective is not just to identify data points, but to extract them into a separate, clean list that can be used for subsequent analysis, charting, or visualization purposes.

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

Once the formula is correctly entered into **C2**, it is applied to the remaining dataset using the drag-and-fill handle down the entirety of Column C. Excel automatically adjusts the cell reference (e.g., changing **B2** to **B3**, **B4**, and so on) for each subsequent row, ensuring that every player's score is tested against the static range of 20 to 30. The resulting column clearly separates the scores that fall within the target range from those that do not, leaving blank entries where the conditional requirements failed.

As clearly visible in the output below, Column C now functions as a highly targeted filter. If the value in Column B is between 20 and 30 (inclusive), the score is returned. Otherwise, the corresponding cell in Column C remains blank. This distinction is powerful for reporting and **data cleaning**, allowing analysts to quickly focus their attention exclusively on the relevant subgroup identified by the conditional logic we established using the IF function.



	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					

Advanced Usage: Returning Custom Text or Boolean Outputs

While returning the original numerical value (data extraction) is highly effective, many scenarios require the analyst only to generate a categorical label indicating whether the criterion was met. Instead of extracting the score itself, we might prefer a simple "Yes" or "No," or a "Pass" or "Fail" designation. This shifts the purpose of the formula from raw data extraction to precise **data classification**, which is particularly valuable for creating summary tables or for use in subsequent conditional formatting rules.

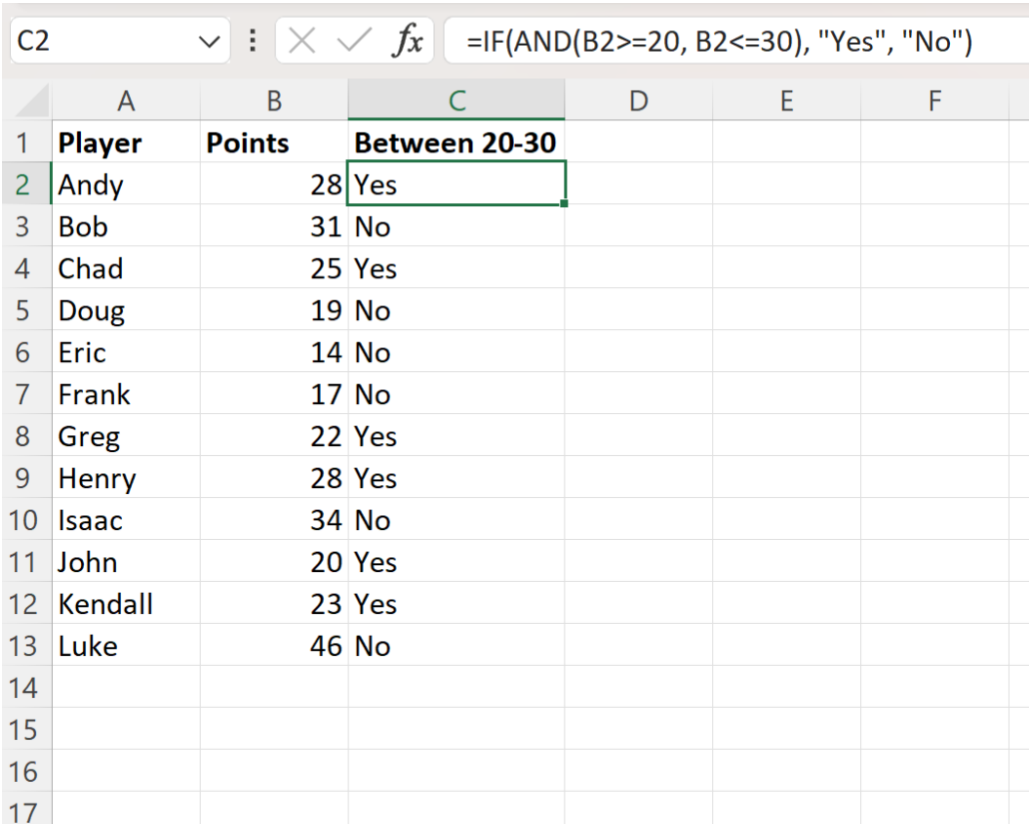
To implement this change, we only need to modify the second and third arguments of the outer IF function. Instead of returning `B2` (the original value) if the condition is TRUE, we insert a descriptive text string, such as `"Yes"`. Similarly, instead of returning `""` (blank) if FALSE, we can insert an alternative text string, such as `"No"`. It is a critical syntax rule to remember that all text strings within a formula must be enclosed in double quotation marks (`""`).

The revised formula below uses this approach to provide clear categorical feedback instantly:

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

As demonstrated in the screenshot below, Column C now provides a definitive "Yes" or "No"

indicator for every row, making it immediately clear which values in Column B fall within the designated range of 20 to 30 points. This method significantly enhances readability when the immediate numerical value is less important than the status of the data point relative to the criteria. This conversion from numerical output to boolean classification is a fundamental step in preparing data for dashboard visualization or summary reporting.



	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						

Understanding Boundary Conditions and Syntax Nuances

When defining numerical ranges using the AND function, careful attention must be paid to the comparison operators, as they fundamentally determine whether the boundaries themselves are included in the valid range. The working examples above utilized `>=` (greater than or equal to) and `<=` (less than or equal to). These operators create an **inclusive range**, meaning a score of exactly 20 or exactly 30 would satisfy the criteria and return TRUE.

However, in certain statistical or business contexts, you may require an **exclusive range**, where the boundary numbers are specifically excluded. For instance, if you wished to find scores strictly greater than 20 and strictly less than 30 (so 20.5 would qualify, but 20 or 30 would not), you must adjust the operators accordingly:

For an inclusive range (20 to 30, including endpoints): `AND(B2>=20, B2<=30)`

For an exclusive range (between 20 and 30, excluding endpoints): `AND(B2>20, B2<30)`

The correct choice of operator is critical, as even a small change can significantly alter the resulting dataset. Furthermore, this nested logic can be combined with other functions, such as the `OR` function (if you needed to check if a value fell into one of two different, non-contiguous ranges) or `COUNTIF/SUMIF` (if you wanted to count or sum all values that meet the criteria without extracting the individual values). Mastering the AND function within the IF function provides the robust foundation necessary for nearly all multi-criteria evaluations in Excel.

Additional Resources for Conditional Logic in Excel

The utilization of nested conditional functions like `IF(AND())` is just one component of powerful spreadsheet analysis. Users interested in further optimizing their data processing skills may wish to explore related tutorials covering other common operations and logical structures within Excel.

These foundational concepts, including absolute vs. relative cell referencing and advanced array formulas, build upon the basic principles demonstrated here, allowing for even more complex data manipulation and reporting across large datasets.

The following resources offer guidance on related conditional and lookup operations:

Using the IF function with multiple criteria (nested IFs).

Implementing VLOOKUP or XLOOKUP for returning values based on specific key matches.

Applying conditional formatting rules based on numerical ranges.

Utilizing the SUMIFS and COUNTIFS functions for aggregated conditional calculations.