How to Find the First Negative Value in a Range

Authored by

stats writer

November 18, 2025

1. Introduction: The Power of Dynamic Range Lookup in Excel

In modern Excel, efficient data analysis often requires techniques that go beyond simple summation or averaging. One common yet crucial task is identifying the precise moment a trend shifts--specifically, locating the **first negative value** within a defined range of data. This scenario is particularly relevant in financial tracking, project management, or quality control, where moving from positive performance to negative results triggers immediate action. Historically, achieving this goal in Excel required complex array formulas involving functions like Array Formula, INDEX, and MATCH.

However, with the introduction of dynamic functions, solving this analytical challenge has become remarkably straightforward and robust. The modern approach utilizes a powerful combination of the XLOOKUP function and the specialized SIGN function. This combination allows us to search through a transformed range of data based on criteria (positive or negative) and return not just the value itself, but any corresponding data in an adjacent column, such as an employee name or transaction date.

This guide will walk you through the construction and application of this efficient formula. We will demonstrate how to structure the logic to accurately pinpoint the transition point from profitability to loss, ensuring that you can always locate the very first instance of negative performance within your dataset quickly and reliably.

2. The Advanced Formula for Negative Value Detection

To efficiently locate the first negative entry in a specified range within Excel, we employ a sophisticated but concise formula. This solution bypasses the need for complex Boolean logic by numerically converting the data points into identifiers that represent their status (positive, negative, or zero).

The general structure of the formula designed to find the first negative value in a range is as follows. Assuming our lookup range is `B2:B13` and we wish to return corresponding data from the range `A2:B13`:

=XLOOKUP(-1,SIGN(B2:B13),A2:B13)

This powerful single-cell formula achieves two critical objectives simultaneously: first, it evaluates every cell in the target range `B2:B13` to determine if it is negative; and second, upon finding the first match, it returns the entire corresponding row's information from the designated return range, `A2:B13`. The magic lies in the core logic: we are essentially telling XLOOKUP to search for the numeric indicator **-1** within a newly generated virtual array.

Understanding why we search for **-1** requires a closer look at the functionality of the SIGN function, which is the engine that converts our raw numerical data into this searchable array of indicators. The application of this formula significantly simplifies tasks that previously required multi-step or legacy Array Formula implementations.

3. Deconstructing the Formula: XLOOKUP and SIGN Functions

The efficacy of this method relies entirely on the interplay between the lookup mechanism (XLOOKUP) and the data transformation tool (SIGN). To master this technique, one must understand how each component contributes to the final result. The formula structure is `XLOOKUP(lookup_value, lookup_array, return_array)`.

The `lookup_value` is statically set to **-1**. This is the target we are searching for, representing the negative status of a number.

The `lookup_array` is dynamically generated by `SIGN(B2:B13)`. This expression evaluates the original data and instantly creates a new virtual array composed entirely of 1s, 0s, and -1s.

The `return_array` is the original range `A2:B13`. This is the set of values that XLOOKUP will pull from once it finds the first match in the virtual lookup array.

The key to its efficiency lies in how XLOOKUP processes data. Unlike older functions which required strict matching modes, XLOOKUP is designed to search through arrays efficiently, and critically, it automatically stops at the very first match it finds. Since Excel reads data top-to-bottom, the first instance of **-1** it encounters in the transformed array directly corresponds to the row containing the first negative value in the original data.

By transforming the raw numerical data into simple status indicators, we drastically simplify the lookup process. For instance, if the cell `B5` contains the value `-150`, the `SIGN(B5)` operation instantly converts it to `-1`. If `B6` contains `200`, it becomes `1`. If `B7` contains `0`, it remains `0`. Searching for the static value **-1** within this binary array is significantly faster and more reliable than searching for a variable value that meets the criteria of being less than zero.

4. Setting Up the Practical Example

To illustrate the application of this formula in a real-world scenario, consider a corporate dataset tracking employee performance, specifically focusing on financial outcomes. We have two columns: one identifying the employee, and another tracking their net profits for a given period. The goal of this data analysis task is to immediately identify which employee first recorded a loss, signaling the need for immediate intervention or review.

Suppose we have the following dataset spanning rows 2 through 13, structured with Employee Names in Column A and Net Profits in Column B. Notice how the values in the **Net Profits** column vary, displaying a mix of positive and negative results across different employees.

The core challenge here is to create a dynamic lookup that doesn't just check if a negative value exists, but specifically retrieves the data associated with the very first occurrence of a loss.

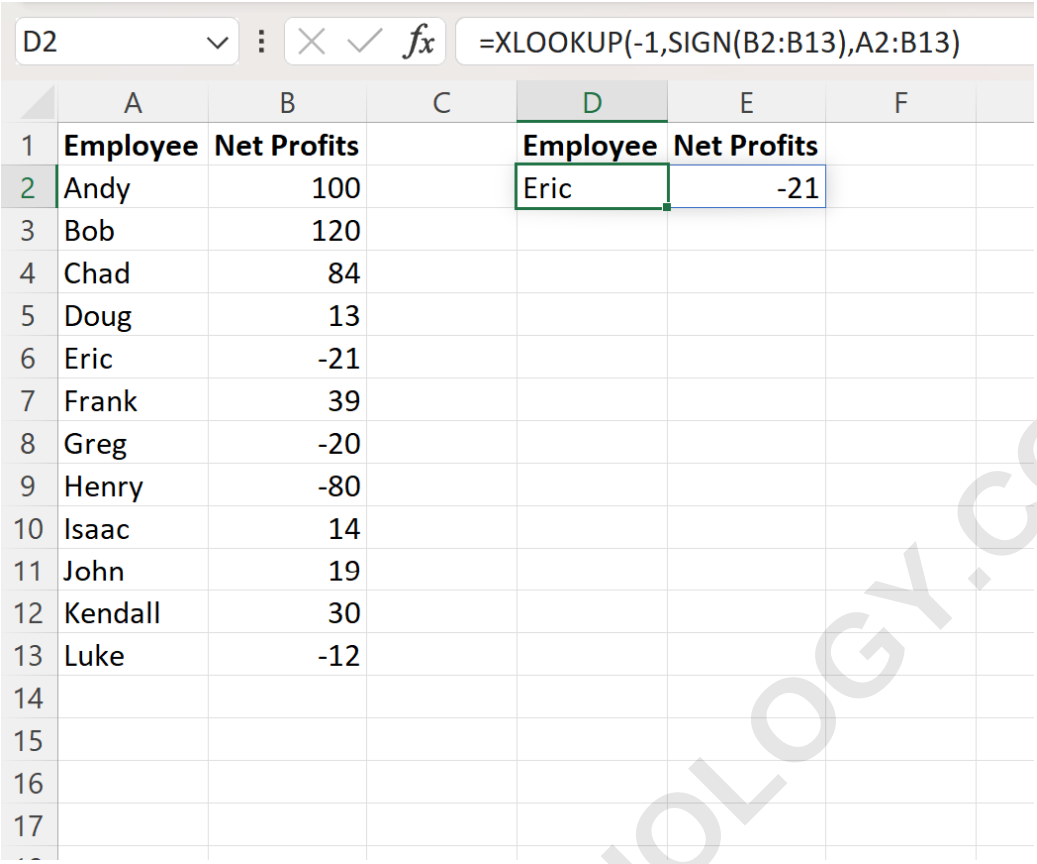
	A	B	C	D	E
1	**Employee**	**Net Profits**			
2	Andy	100			
3	Bob	120			
4	Chad	84			
5	Doug	13			
6	Eric	-21			
7	Frank	39			
8	Greg	-20			
9	Henry	-80			
10	Isaac	14			
11	John	19			
12	Kendall	30			
13	Luke	-12			
14					
15					
16					
17					

Our primary target range for inspection is `B2:B13` (the Net Profits), and our desired output range is `A2:B13` (both Employee Name and Net Profits). We will place our result formula in cell `D2` for clear visibility.

5. Applying the Solution: Returning the Full Record

To execute the lookup and return the complete record corresponding to the first loss, we enter the core formula directly into cell `D2`. We instruct Excel to look up the indicator for negative status (`-1`) within the transformed array derived from the profits column (`SIGN(B2:B13)`), and upon finding it, return the corresponding data from the Employee and Net Profits columns (`A2:B13`).

The exact formula typed into cell D2 is: `=XLOOKUP(-1,SIGN(B2:B13),A2:B13)`. Once entered, the formula dynamically spills the result across multiple cells, identifying the employee name and the negative profit value associated with that first instance.



	A	B	C	D	E	F
1	**Employee**	**Net Profits**		**Employee**	**Net Profits**	
2	Andy	100		Eric	-21	
3	Bob	120				
4	Chad	84				
5	Doug	13				
6	Eric	-21				
7	Frank	39				
8	Greg	-20				
9	Henry	-80				
10	Isaac	14				
11	John	19				
12	Kendall	30				
13	Luke	-12				
14						
15						
16						
17						

As shown in the output, the formula successfully identifies "Eric" with a Net Profit of -$150. This confirms that Eric is the first employee in the list, reading from top to bottom, whose profit figure dipped below zero. This immediate verification is crucial for maintaining data integrity and ensuring the formula executes as intended.

We can manually verify this result by scanning the **Net Profits** column (B2:B13). The values for Ann (B2), Bob (B3), Cathy (B4), and David (B5) are all positive. The value for Eric (B6) is -$150, marking the transition point. Since `SIGN` converts this value to -1, and XLOOKUP seeks the first -1, the result is robustly linked to the correct row.

	A	B	C	D	E
1	**Employee**	**Net Profits**		**Employee**	**Net Profits**
2	Andy	100		Eric	-21
3	Bob	120			
4	Chad	84			
5	Doug	13			
6	**Eric**	**-21**			
7	Frank	39			
8	Greg	-20			
9	Henry	-80			
10	Isaac	14			
11	John	19			
12	Kendall	30			
13	Luke	-12			
14					
15					
16					

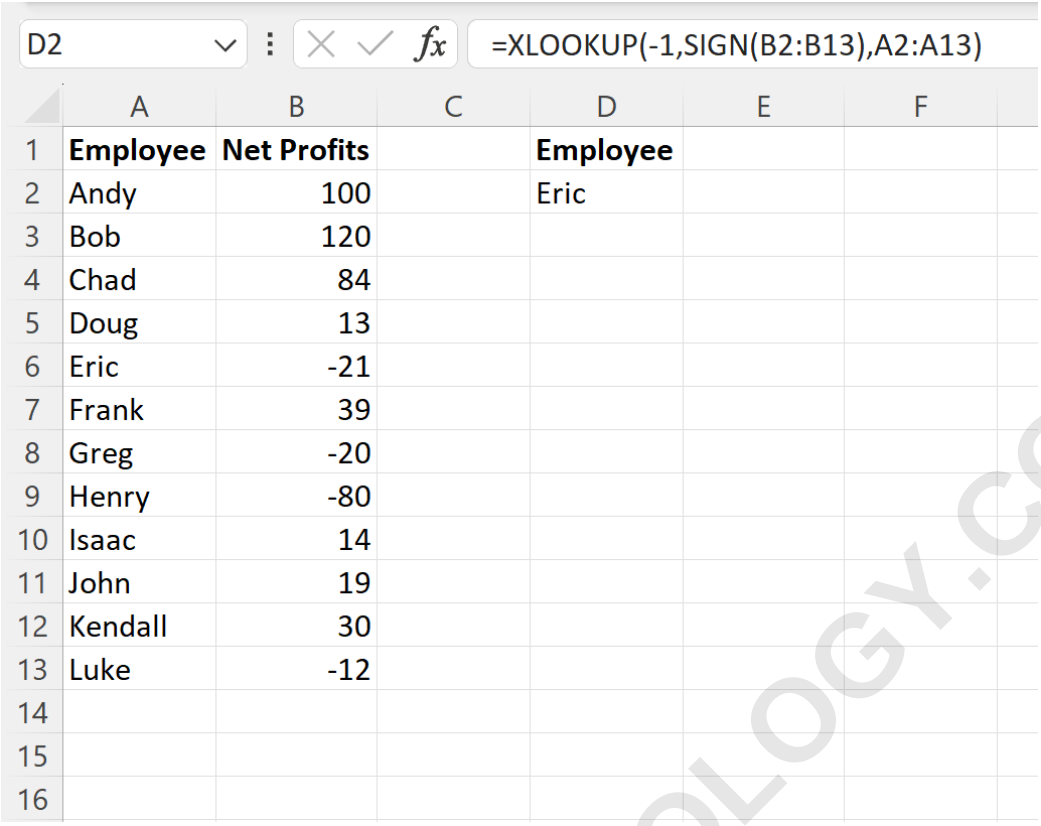
6. Refining the Output: Returning Specific Columns

While returning the full record (both Employee and Net Profits) provides comprehensive information, there are scenarios where only a single data point--such as the employee's name--is required for reporting or further lookups. The beauty of the XLOOKUP function is its flexibility in defining the `return_array` argument.

If we only need the name of the employee responsible for the first negative profit, we must adjust the final argument of the formula from `A2:B13` to focus exclusively on the column containing the employee names, which is `A2:A13`. The refined formula placed in cell `D2` becomes:

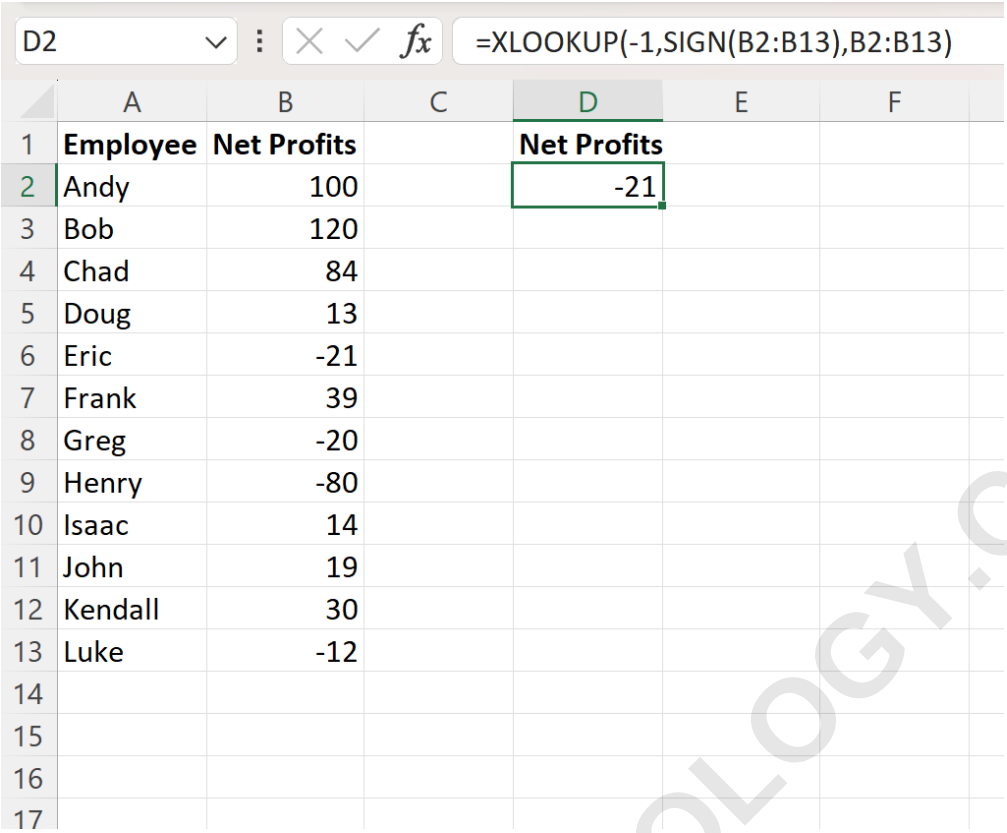
=XLOOKUP(-1,SIGN(B2:B13),A2:A13)

By making this subtle but crucial modification to the `return_array`, the lookup process remains identical--it still searches the `SIGN(B2:B13)` array for the first **-1**. However, instead of retrieving values from the two columns, it only extracts the corresponding employee name from column A. This demonstrates the power of the dynamic array capabilities in modern Excel, allowing for precise control over the output format based on the business requirement.



	A	B	C	D	E	F
1	**Employee**	**Net Profits**		**Employee**		
2	Andy	100		Eric		
3	Bob	120				
4	Chad	84				
5	Doug	13				
6	Eric	-21				
7	Frank	39				
8	Greg	-20				
9	Henry	-80				
10	Isaac	14				
11	John	19				
12	Kendall	30				
13	Luke	-12				
14						
15						
16						

The result of this modification is a cleaner output, returning only "Eric" in cell D2, confirming his position as the first employee to record a loss, without displaying the profit figure itself. This selective retrieval is highly beneficial when integrating this result into other summary tables or dashboards.



	A	B	C	D	E	F
1	**Employee**	**Net Profits**		**Net Profits**		
2	Andy	100		-21		
3	Bob	120				
4	Chad	84				
5	Doug	13				
6	Eric	-21				
7	Frank	39				
8	Greg	-20				
9	Henry	-80				
10	Isaac	14				
11	John	19				
12	Kendall	30				
13	Luke	-12				
14						
15						
16						
17						

7. Understanding the Mechanism: How SIGN Transforms Data

A thorough understanding of the `SIGN` function (or Signum function) is essential to appreciate the elegance of this lookup technique. The `SIGN` function is a mathematical operator in Excel designed specifically to return the sign of a number, making it perfectly suited for converting large datasets into a lookup-friendly format based on numerical status.

When applied to a range, `SIGN(Range)` converts every numerical entry into one of three distinct integer outcomes:

1 (Positive): If the number in the cell is greater than zero (e.g., 500 becomes 1).

-1 (Negative): If the number in the cell is less than zero (e.g., -150 becomes -1).

0 (Zero): If the number in the cell is exactly zero (e.g., 0 becomes 0).

In the context of our formula, `SIGN(B2:B13)` generates a virtual array like `{1; 1; 1; 1; -1; 1; 1; 0; -1; 1; 1; 1}`. It is into this specific sequence that the `XLOOKUP` function searches for the target **-1**. Because this process creates a one-to-one mapping between the status indicator array and the original data array, the position of the first **-1** in the virtual array perfectly aligns with the position of the first negative value in the original **Net Profits** column.

This method is superior to using conditional formatting or basic filters, as it provides a dynamic, self-updating output in a single cell, integrated seamlessly into the spreadsheet workflow. It eliminates the ambiguity of searching for "less than zero" across an entire range and replaces it with the precise search for the integer **-1**.

8. Addressing Edge Cases and Limitations

While the `XLOOKUP` and `SIGN` combination is highly effective, it is important to consider edge cases that might affect the result. One potential issue involves the presence of zero values. As `SIGN(0)` returns `0`, the formula correctly ignores zero entries, focusing only on positive (1) or negative (-1) values. This is ideal when searching strictly for a loss.

Another consideration is the treatment of errors or non-numeric text within the lookup array (e.g., `B2:B13`). If the range contains text or error values, the `SIGN` function will return an error for that cell position, propagating an error into the `XLOOKUP` function. To handle this robustly, one might wrap the `SIGN` component with an error-handling function like `IFERROR` or `IFNA`, or utilize the optional fourth argument of `XLOOKUP` (if_not_found). A common refinement for professional data sets is `=XLOOKUP(-1, SIGN(IFERROR(B2:B13, 1)), A2:B13)`, which temporarily treats any error or non-numeric entry as a positive value (1), ensuring the lookup continues past the corrupt data point until a true negative (-1) is found.

Furthermore, unlike older formulas that required Ctrl+Shift+Enter to function as Array Formula, the `XLOOKUP` function is inherently dynamic and automatically handles array operations without special keystrokes, significantly enhancing user experience and formula readability. This modern approach is highly recommended for all data analysis tasks in contemporary versions of Excel.

9. Conclusion

Identifying the first instance of a negative value in a large dataset is a fundamental analytical requirement, crucial for timely decision-making. By leveraging the combined power of the `XLOOKUP` and `SIGN` functions, Excel users can create dynamic, robust, and highly readable formulas that accomplish this task with unparalleled efficiency.

Whether you choose to return the full record or just a specific identifier, the methodology remains the same: transform the data into searchable indicators (1, 0, -1) and search for the indicator corresponding to the negative status (-1). This approach not only solves the immediate problem but also lays the groundwork for more complex conditional lookups based on numeric status rather than specific values.

Mastering this formula ensures that your spreadsheet models are equipped with the best tools for modern, array-aware computation, allowing for faster insights and reduced reliance on legacy

lookup methods.