Excel: Find First 3 Positive Numbers in Column

Authored by

Mohammed looti

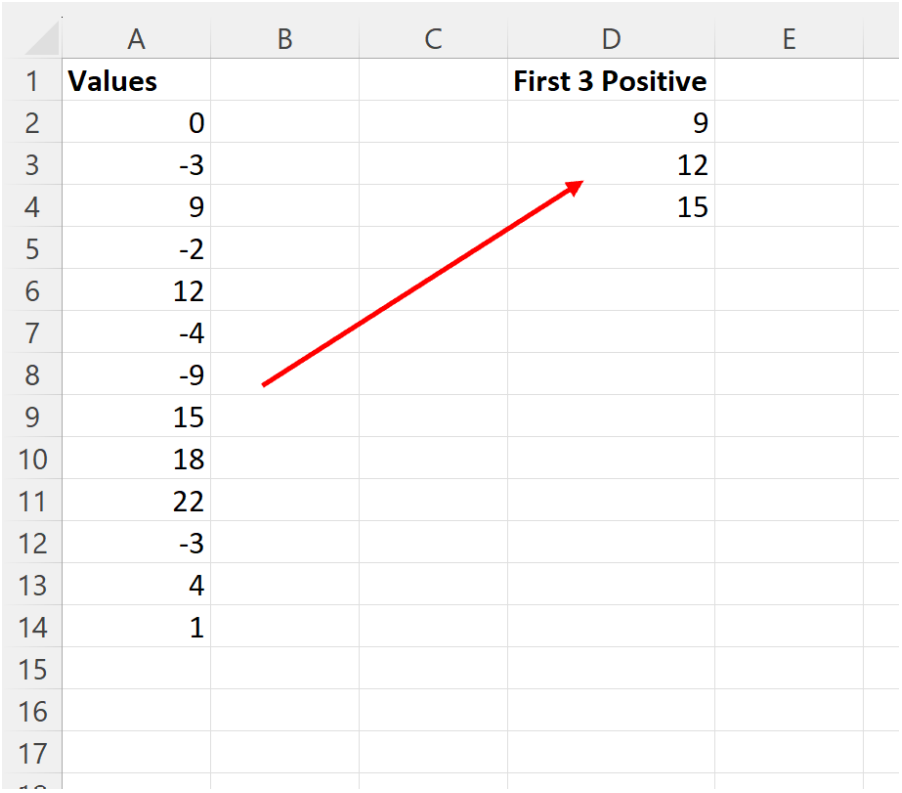
November 10, 2025

The Challenge of Sequential Data Filtering in Excel

In advanced data analysis using Excel, practitioners often face a unique challenge: extracting data based on a specific condition while strictly maintaining the original order of appearance. This requirement goes beyond simple filtering or sorting. Analysts frequently need to isolate the first few instances--in this case, the first three positive numbers--exactly as they appear sequentially within a column. Standard tools like auto-filters or pivot tables are inadequate for this task because they typically rearrange the data, thereby destroying the positional context crucial for time-series or deviation tracking.

Imagine a scenario where you are monitoring performance metrics or tracking operational deviations over time. To understand the initial trigger points, you must isolate the first, second, and third instances where the change registered as positive, based strictly on the row number. The methodology must be robust enough to ignore negative values and zeros without losing track of the sequence of the remaining data points.

For example, given a column containing a mix of positive and negative figures, our goal is to return only the first three positive occurrences encountered when scanning the list from top to bottom, as illustrated below:



	A	B	C	D	E
1	**Values**			**First 3 Positive**	
2	0			9	
3	-3			12	
4	9			15	
5	-2				
6	12				
7	-4				
8	-9				
9	15				
10	18				
11	22				
12	-3				
13	4				
14	1				
15					
16					
17					

To achieve this complex sequential extraction in Excel, we must deploy a sophisticated

combination of functions, utilizing the power of an Array Formula to implement conditional indexing. The following detailed guide breaks down the precise methodology required to execute this highly specific data retrieval technique.

Setting Up the Data and Understanding the Logic

To properly demonstrate this advanced filtering technique, we will establish a sample dataset in an Excel worksheet. This dataset will contain a variety of integers, including positive values, negative values, and zero. Our primary objective is to create a formula that can successfully isolate only the positive figures while rigorously preserving their original order relative to the start of the list.

Consider the following list of values populating Column A, specifically ranging from cell **A2** to **A14**, which serves as our source data:

	A	B	C	D	E
1	**Values**				
2	0				
3	-3				
4	9				
5	-2				
6	12				
7	-4				
8	-9				
9	15				
10	18				
11	22				
12	-3				
13	4				
14	1				
15					
16					
17					
18					

As you observe the data, note the mixture of data types: we have multiple positive numbers (9, 12, 15, 25, 33), several negative values (-5, -11), and a zero (0). The core logic necessary for this solution involves two steps: first, identifying the row number of every entry that meets the positive criteria, and second, sequentially pulling the value associated with the first three of those identified row numbers. This sequential ranking and extraction is efficiently handled by combining conditional

logic with the powerful ranking capabilities provided by the SMALL function.

Implementing the Sequential Indexing Helper Column

Before we can construct the comprehensive Array Formula, we require a simple, yet vital, helper mechanism. This mechanism instructs the formula exactly which positive number it should retrieve--the 1st, 2nd, or 3rd. This is achieved by listing the integers 1 through 3 in an adjacent column, which will act as the crucial 'k' value, or rank, for our subsequent functions.

To find the first three positive values, we must list the numbers 1 through 3 in column **C**, starting in cell **C1**:

	A	B	C	D	E	F
1	**Values**		1			
2	0		2			
3	-3		3			
4	9					
5	-2					
6	12					
7	-4					
8	-9					
9	15					
10	18					
11	22					
12	-3					
13	4					
14	1					
15						
16						
17						

This dedicated helper column (Column C) is indispensable. The final formula will dynamically reference **C1** when seeking the first positive value, **C2** for the second, and **C3** for the third. By using this dynamic, relative reference, the formula can be copied down effortlessly, automating the sequential search process. Attempting this complex task without this ranking input would result in the formula simply returning the first positive number repeatedly, failing the sequential requirement.

Constructing the Core Array Formula

The core of this elegant solution resides in a single, powerful Array Formula. It skillfully integrates three essential Excel functions: INDEX, IF, and SMALL. We will enter this formula into cell **D1** to initiate the search for the first positive value in Column A.

Ensure the formula is entered precisely into cell **D1**. It is important to recall that in traditional versions of Excel, this complex formula requires confirmation by pressing **CTRL + SHIFT + ENTER** simultaneously to activate its array processing capabilities (often visualized by curly braces around the formula). However, users of modern Excel environments featuring Dynamic Arrays may find this manual confirmation unnecessary.

=INDEX($A$2:$A$14,SMALL(IF($A$2:$A$14>0,ROW($A$2:$A$14)-ROW($A$2)+1),C1))

To fully appreciate this method, let us dissect the logic of this powerful expression step-by-step, understanding how it pinpoints the exact position of the required sequential value:

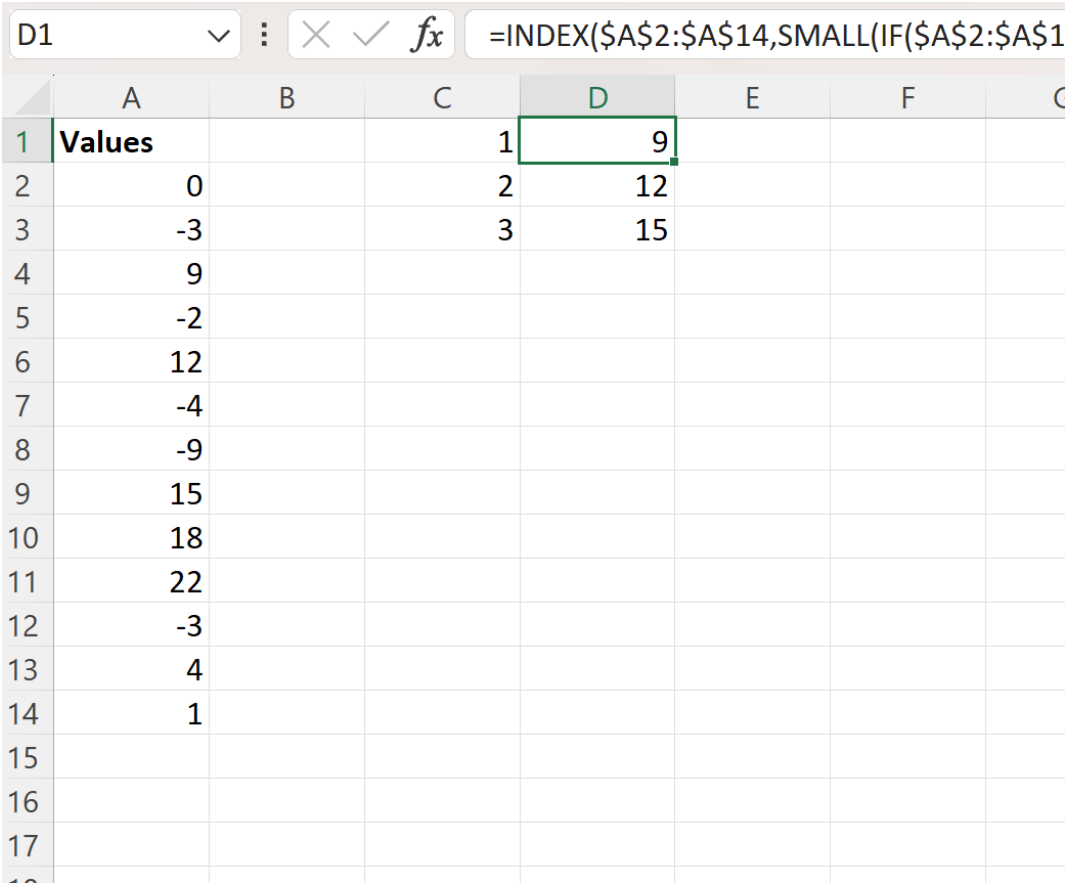
The Conditional Filter (IF($A$2:$A$14>0, ...)): This initial expression evaluates every cell in the defined range (`$A$2:$A$14`). It generates a logical array composed of TRUE and FALSE flags, effectively marking every entry that satisfies the criteria (is greater than zero) as TRUE.
Generating Relative Row Numbers (ROW($A$2:$A$14)-ROW($A$2)+1): This crucial segment calculates the relative position of each row within the specified data range. It skillfully converts the absolute sheet row number (e.g., Row 2, 3, 4) into a clean, relative index starting at 1 (i.e., 1, 2, 3...). This transformation is essential because the INDEX function requires an index number relative to the start of its array, not the absolute row on the sheet.
Filtering the Indices: The `IF` function then uses the TRUE/FALSE array to filter the relative row indices. It returns the index only for those cells where the condition (value > 0) is TRUE. For all non-positive values, the IF function effectively omits the row number. The resulting output is an array containing only the relative row indices of all positive values.
Finding the Kth Smallest Index (SMALL(..., C1)): The SMALL function is used next to extract the Kth smallest number from this filtered array of indices. The value of K is determined by the ranking counter in cell **C1** (which is 1 for the first result). This step precisely identifies the relative position of the first positive number in the sequence.
Returning the Final Value (INDEX($A$2:$A$14, ...)): Finally, the INDEX function uses the positional index number returned by the SMALL function to pull the actual corresponding value from the source data range `$A$2:$A$14`.

Extracting the Sequential Results

Once the complex formula is correctly entered into cell **D1** (and confirmed as an array if necessary), extracting the subsequent positive values becomes a straightforward process. This efficiency is possible because we diligently used absolute references (e.g., `$A$2:$A$14`) for the

data range, ensuring the range never shifts, but maintained a relative reference (`C1`) for the rank counter. This design allows for seamless copying.

We simply click and drag the formula down to cells **D2** and **D3**. When the formula moves to **D2**, its reference automatically adjusts to **C2** (which contains the value 2). This instructs the SMALL function to find the second smallest index, thereby returning the second positive value in the sequence. Similarly, **D3** references **C3**, accurately retrieving the third sequential positive value.



	A	B	C	D	E	F
1	**Values**		1	9		
2	0		2	12		
3	-3		3	15		
4	9					
5	-2					
6	12					
7	-4					
8	-9					
9	15					
10	18					
11	22					
12	-3					
13	4					
14	1					
15						
16						
17						

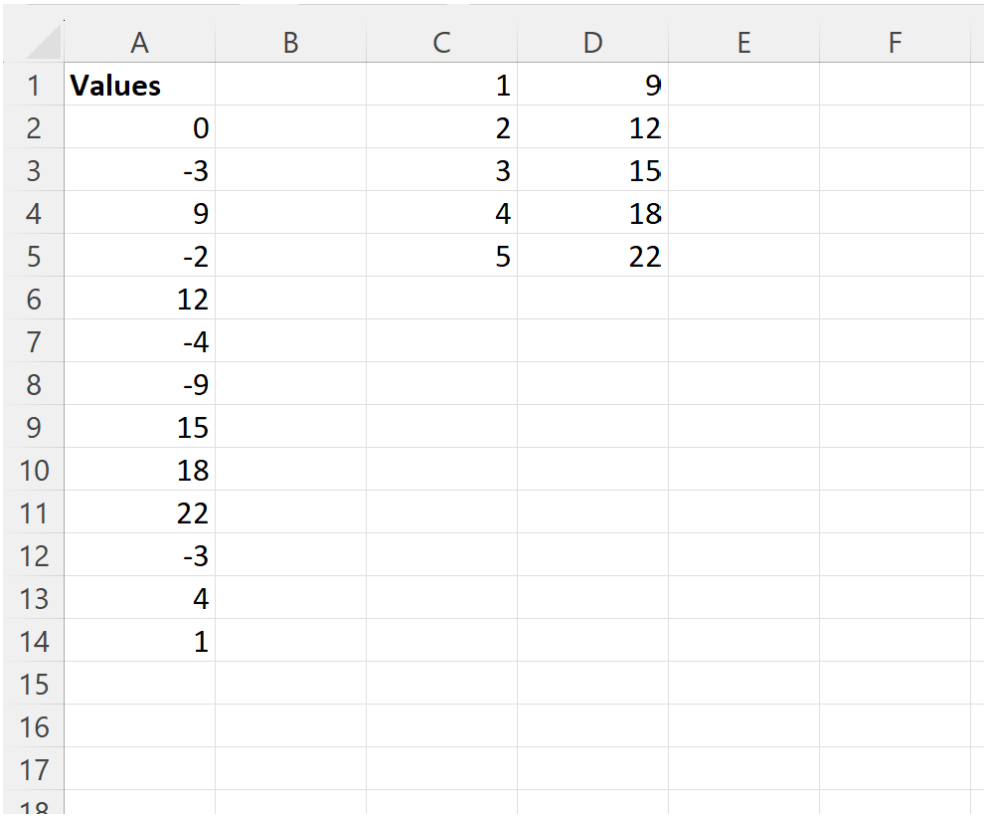
As clearly demonstrated in the resulting spreadsheet view, Column D now accurately displays the first three positive values extracted from Column A, retaining their exact order of appearance: **9**, **12**, and **15**. Crucially, later positive values (such as 25 and 33) were correctly skipped because they did not meet the rank criteria, confirming that the array logic successfully prioritized sequential position over the magnitude of the number.

Generalizing the Solution for N Positive Values

A significant advantage of employing this dynamic Array Formula method is its inherent scalability. The logic is not confined to finding only the first three values; it can be readily adapted to find the first *n* positive numbers in any column, provided that the sequential helper column (Column C) is

extended appropriately to define the desired rank.

For instance, if your analytical requirement shifted to identifying the first five instances of positive change, you would simply extend Column C to include the integers 1 through 5. You could then drag our established formula down to cells **D4** and **D5** to list out the fourth and fifth positive values from Column A:



	A	B	C	D	E	F
1	**Values**		1	9		
2	0		2	12		
3	-3		3	15		
4	9		4	18		
5	-2		5	22		
6	12					
7	-4					
8	-9					
9	15					
10	18					
11	22					
12	-3					
13	4					
14	1					
15						
16						
17						

The updated Column D now accurately displays the first five positive values from the source data: 9, 12, 15, 25, and 33. It is important to note the error handling: if the formula is dragged down further than the total number of positive values available in Column A, it will return the standard Excel error **#NUM!**. This error serves as a clear indicator that no more positive values exist to satisfy the specified rank. Therefore, matching the length of the helper column (C) to the maximum number of expected or available results is a best practice. This highly versatile formula can be adapted for any dataset where sequential, conditional extraction is a necessity.

Summary of Key Excel Functions for Indexing

The successful implementation of this advanced data extraction technique relies fundamentally on the proper interaction of several specialized Excel functions working in concert within the array structure. A solid grasp of each component's specific role is crucial for adapting the solution to different filtering criteria, such as finding negative numbers or isolating values above a custom

threshold.

Here is a brief, functional overview of the key components central to this array logic:

INDEX: This function is responsible for the final output. It returns the value of a cell at a specified position within a defined range. In this context, it fetches the actual data point (the positive number) after its relative row position has been precisely determined.
IF: Serving as the conditional filter, the IF function determines which data points meet the criteria (value greater than zero). Critically, it returns the positional index only for those data points that satisfy the condition, filtering out all others.
SMALL: This function retrieves the Kth smallest value from a specified array of numbers. In our array formula, it is used to pull the 1st, 2nd, or 3rd smallest *relative row index* returned by the IF function, effectively ranking the positive numbers by their original appearance order.
ROW: This function is used to generate the necessary sequence of numerical indices (1, 2, 3...) that accurately represent the positions of the data within the defined range. This ensures the INDEX function retrieves the correct corresponding value based on its position within the source data.

Additional Resources for Advanced Excel Operations

Mastering sequential filtering techniques using complex conditional Array Formula structures is an indispensable step toward achieving advanced proficiency in Excel. The ability to combine logical operations and lookup functions into robust array structures is essential for efficiently handling nearly any complex dataset.

The following tutorials explain how to perform other common and advanced operations in Excel, expanding on the concepts of conditional formatting and data extraction methods discussed here:

How to Find the Last Positive Value in a Column.
Using the AGGREGATE Function for Complex Filtering.
Understanding Dynamic Array Formulas in Modern Excel.