How to Find the First Number in a Text String Using Excel

Authored by

stats writer

February 20, 2026

Effective **data management** within **Microsoft Excel** often requires the ability to parse complex strings that contain a mixture of alphabetic characters and numerical digits. Whether you are dealing with inconsistent product codes, serial numbers, or **alphanumeric** employee IDs, isolating the first numeric digit is a fundamental step in **data cleaning**. This process allows users to transform raw, unstructured text into organized information that can be sorted, filtered, and analyzed with precision. By mastering specific text functions, you can automate what would otherwise be a tedious manual task, significantly increasing your productivity in any **data analysis** workflow.

Excel: Find First Number in Text String

Understanding the Core Logic of Text Parsing

To successfully identify the first number within a string, we must leverage the power of the **FIND function** in **Microsoft Excel**. The **FIND function** is designed to locate the starting position of a specific substring within a larger body of text. However, when the goal is to find "any" number rather than a specific one, the standard application of the formula requires a more sophisticated approach. We achieve this by using an **array constant**, which allows the function to search for multiple potential matches--specifically the digits 0 through 9--simultaneously.

The complexity of **alphanumeric** strings often stems from the fact that numbers can appear at any position. In a dataset of employee identifiers, one code might start with three letters followed by numbers, while another might start with a single digit. This variability makes static functions like **LEFT function** or RIGHT insufficient on their own. Instead, we must create a dynamic formula that scans the entire length of the cell to pinpoint exactly where the numeric sequence begins.

Furthermore, we must account for the possibility that a string might not contain any numbers at all. In professional **data analysis**, a robust formula should be resilient to such edge cases. By combining the **FIND function** with the **MIN function**, we can evaluate the positions of all ten possible digits and select the lowest numerical value, which corresponds to the first occurrence. This layered logic ensures that the user receives an accurate result regardless of the string's internal structure.

The Mechanics of the FIND and MIN Functions

The primary formula used to locate the position of the first digit utilizes a clever **concatenation** trick. By appending the string "0123456789" to the end of the cell reference, we guarantee that the **FIND function** will always find every digit it is searching for. This prevents the formula from returning a **#VALUE! error** if a specific digit is missing from the original text. The **MIN function** then identifies the smallest position found, which will always be the first number in the actual cell

content unless no numbers exist in the cell, in which case it points to the appended suffix.

When you use an **array formula** approach within the **FIND function**, Excel treats the set of numbers `{0,1,2,3,4,5,6,7,8,9}` as individual search queries. For each digit in that array, Excel calculates its earliest position in the modified string. This results in a temporary array of ten position values. The **MIN function** then scans this internal list to extract the lowest number, effectively pinpointing the "index" of the first digit encountered in the text.

This method is highly efficient for **data cleaning** because it operates within a single cell and does not require complex **VBA** macros or external scripts. It is a native solution that remains compatible across various versions of **Microsoft Excel**, including Excel 365, Excel 2021, and older iterations. Understanding this logic is crucial for anyone looking to build advanced **spreadsheet** models that handle variable text inputs.

You can use the following formulas in Excel to find the first number in a text string:

Formula 1: Return Position of First Number

=MIN(FIND({0,1,2,3,4,5,6,7,8,9},A2&"0123456789"))

Breaking Down the Position Retrieval Formula

The formula provided above is a masterpiece of **concatenation** and logic. Let us break down the components of Formula 1. The inner part, `A2&"0123456789"`, takes the content of cell A2 and attaches the full sequence of digits to the end. If A2 contains "Part#45", the string becomes "Part#450123456789". This ensures that when the **FIND function** looks for the digit '0', it finds it at the end even if it isn't in the original "Part#45".

The **FIND function** then searches for each digit in the array `{0,1,2,3,4,5,6,7,8,9}` within that long string. In our "Part#45" example, the digit '4' is found at position 6, and the digit '5' is found at position 7. Other digits like '0', '1', and '2' are found much later in the string (in the appended section). The **MIN function** evaluates all these positions and identifies that '6' is the smallest value. Therefore, the formula correctly reports that the first number in the string starts at the 6th character.

This approach is vital for maintaining **data integrity**. Without the appended "0123456789", if you searched for a digit that didn't exist in the cell, the **FIND function** would fail for that specific digit, potentially breaking the entire calculation. By providing a "fallback" for every digit, the formula remains robust and reliable for diverse **alphanumeric** datasets.

This formula returns the position of the first number in a string.

For example, if the string is **A0095B** then this formula will return **2** since this is the position in the string where the first number occurs.

Formula 2: Return Value of First Number

=MID(A2,MIN(FIND({0,1,2,3,4,5,6,7,8,9},A2&"0123456789")),1)

Extracting the Actual Numeric Value

While knowing the position of a number is helpful, often the ultimate goal is to extract the actual digit itself. This is where the **MID function** becomes indispensable. The **MID function** allows you to extract a substring from the middle of a text string, provided you know the starting point and the number of characters you wish to retrieve. By nesting our position formula inside the **MID function**, we can automate the extraction of the first digit.

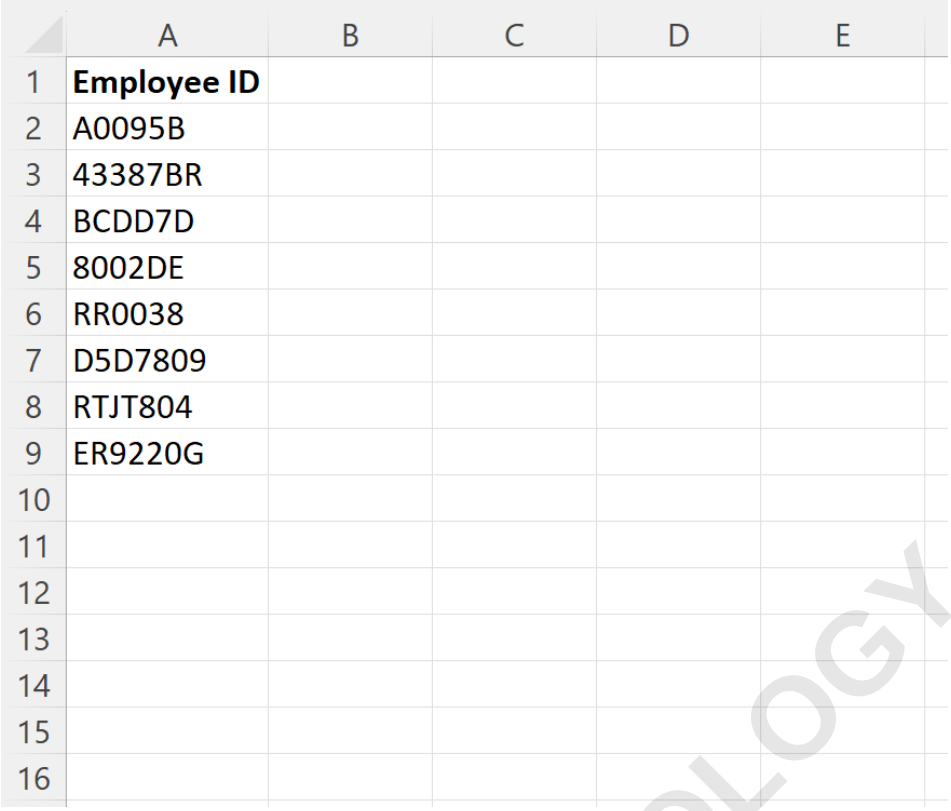
In Formula 2, the **MID function** takes three arguments: the text source (A2), the start position (calculated by our **MIN** and **FIND** logic), and the number of characters (which we set to 1). This tells Excel: "Go to cell A2, find the position where the first number starts, and give me exactly one character from that point." This effectively isolates the first numeric value from the rest of the **alphanumeric** string.

This technique is particularly useful in **data analysis** scenarios where numbers represent categories or priority levels. For instance, if an invoice code starts with a letter followed by a priority digit (e.g., "X104"), extracting the '1' allows you to group data by priority without the interference of the leading letter. By automating this with a formula, you ensure that as your **Microsoft Excel** sheet grows, your extracted data updates in real-time.

This formula returns the value of the first number in a string.

For example, if the string is **A0095B** then this formula will return **0** since this is the value of the first number that occurs in the string.

The following example shows how to use each formula in practice with the following list of employee ID strings in Excel:



	A	B	C	D	E
1	**Employee ID**				
2	A0095B				
3	43387BR				
4	BCDD7D				
5	8002DE				
6	RR0038				
7	D5D7809				
8	RTJT804				
9	ER9220G				
10					
11					
12					
13					
14					
15					
16					

Example 1: Practical Application of Position Detection

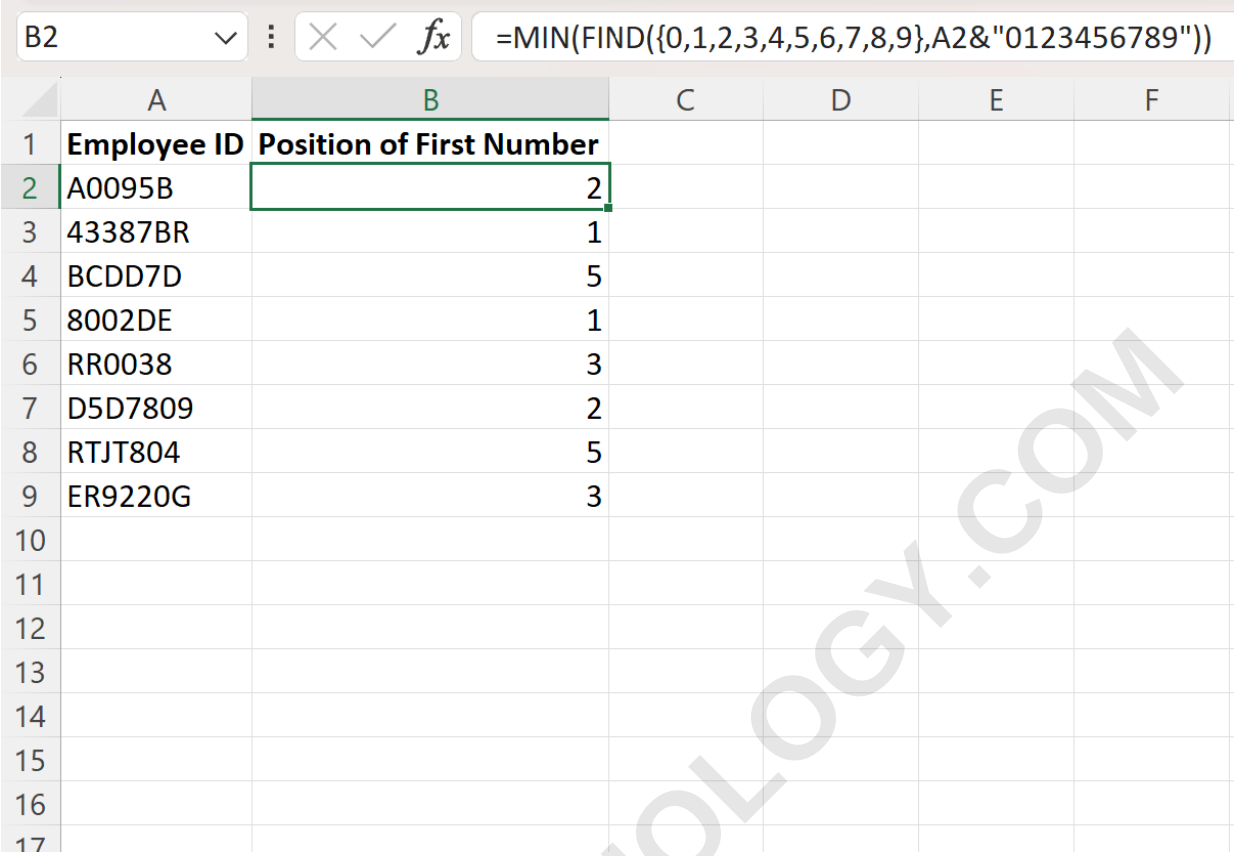
To implement the position detection formula in a real-world scenario, consider a list of Employee IDs located in column A. These IDs are formatted inconsistently, with some having letters at the beginning and others starting with numbers. To identify the exact point where the numeric component begins, we navigate to cell B2 and enter the **MIN** and **FIND** combination formula. This allows us to establish a baseline for further **data cleaning** steps.

Once the formula is entered into the first cell, the **AutoFill** feature in **Microsoft Excel** can be used to apply the logic across the entire column. By clicking and dragging the fill handle, the formula adjusts its relative references, analyzing each subsequent Employee ID in column A. This provides a clear, numerical index for every row, indicating where the transition from text to digits occurs.

We can type the following formula into cell **B2** to return the position of the first number in each Employee ID text string:

=MIN(FIND({0,1,2,3,4,5,6,7,8,9},A2&"0123456789"))

We can then click and drag this formula down to each remaining cell in column B:



	A	B	C	D	E	F
1	**Employee ID**	**Position of First Number**				
2	A0095B	2				
3	43387BR	1				
4	BCDD7D	5				
5	8002DE	1				
6	RR0038	3				
7	D5D7809	2				
8	RTJT804	5				
9	ER9220G	3				
10						
11						
12						
13						
14						
15						
16						
17						

As observed in the resulting **spreadsheet**, Column B now serves as a roadmap for our **alphanumeric** strings. For an ID like "BCDD7D", the formula returns 5, signifying that the first four characters are non-numeric. Conversely, for an ID like "43387BR", the formula returns 1, showing that the string begins immediately with a numeric value. This level of detail is essential for complex **data analysis** tasks where string structure dictates the processing logic.

The first number in **A0095B** occurs in position **2** of the string.
The first number in **43387BR** occurs in position **1** of the string.
The first number in **BCDD7D** occurs in position **5** of the string.

Example 2: Isolating the First Digit for Analysis

In our second example, we shift our focus from finding the position to extracting the actual value. Using the **MID function** in conjunction with our position formula, we can populate column B with the specific first digit of each Employee ID. This is a common requirement in **data analysis** when the first digit of a code represents a specific department, location, or year of entry.

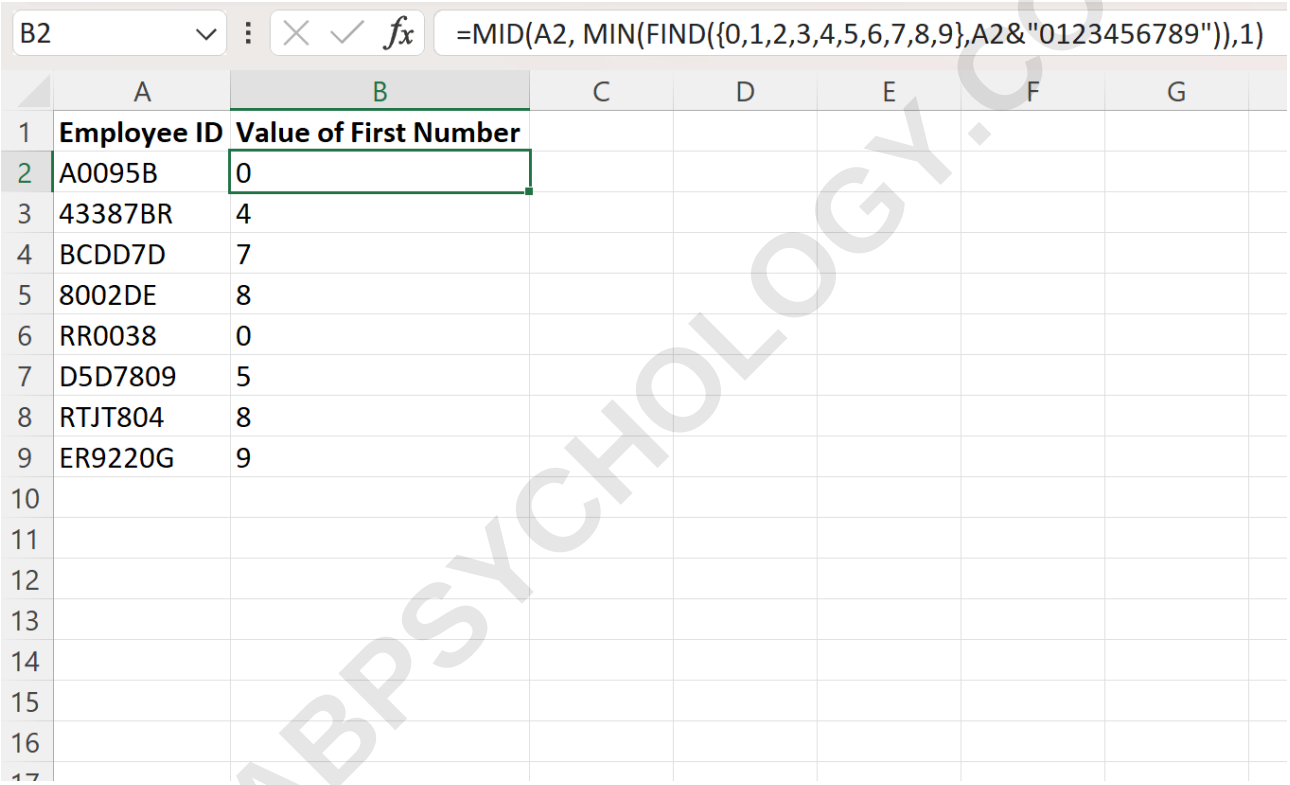
By entering the nested **MID** formula into cell B2, we tell Excel to look at cell A2, find the position of the first number, and pull that single character. The results are immediate and accurate. This process transforms a messy column of mixed text into a refined column of digits that are ready for

Pivot Table analysis or conditional formatting. It highlights the versatility of **Microsoft Excel** in handling non-standard data structures.

We can type the following formula into cell **B2** to return the value of the first number in each Employee ID text string:

=MID(A2,MIN(FIND({0,1,2,3,4,5,6,7,8,9},A2&"0123456789")),1)

We can then click and drag this formula down to each remaining cell in column B:



	A	B	C	D	E	F	G
1	**Employee ID**	**Value of First Number**					
2	A0095B	0					
3	43387BR	4					
4	BCDD7D	7					
5	8002DE	8					
6	RR0038	0					
7	D5D7809	5					
8	RTJT804	8					
9	ER9220G	9					
10							
11							
12							
13							
14							
15							
16							

The output in column B confirms the success of the **data cleaning** operation. For "A0095B", the value extracted is 0. For "BCDD7D", the value is 7. This precision allows users to verify that their **alphanumeric** strings follow expected patterns and to quickly identify outliers that may require further manual investigation.

The value of the first number in **A0095B** is **0**.
The value of the first number in **43387BR** is **4**.
The value of the first number in **BCDD7D** is **7**.

Advanced Considerations and Workflow Optimization

When working with massive datasets in **Microsoft Excel**, efficiency is paramount. While the **FIND** and **MIN** combination is powerful, users should be aware of the difference between **FIND** and **SEARCH**. The **FIND function** is case-sensitive, which is not usually an issue when searching for numbers, but the **SEARCH function** can be used as an alternative if case-insensitivity is required for other parts of a nested formula. However, for **alphanumeric** digit detection, **FIND** remains the industry standard.

To further enhance your **data analysis** workflow, consider wrapping these formulas in an **IFERROR function**. While the concatenation of "0123456789" prevents most errors, an **IFERROR** wrapper can provide a custom message--such as "No Number Found"--when the **MIN** result points to a position beyond the actual length of the original string. This makes your **spreadsheet** more user-friendly for colleagues who may not be familiar with the underlying logic.

Finally, as **array formulas** become more dynamic in modern Excel (Excel 365), these traditional methods continue to be relevant due to their backward compatibility. Whether you are performing a one-time **data cleaning** task or building a permanent dashboard, mastering the ability to navigate and manipulate **alphanumeric** text is a hallmark of an advanced Excel user. This skill set forms the foundation for more complex operations, such as extracting entire numeric sequences or calculating sums from within text blocks.

The following tutorials explain how to perform other common tasks in Excel: