How do you find the first letter in a string in Excel?

Authored by

stats writer

November 18, 2025

Introduction: The Challenge of Extracting Text from Mixed Strings in Excel

Microsoft Excel is an incredibly versatile tool, but manipulating string data that combines both numerical and alphabetical characters often presents a complex challenge. Unlike databases that might have specialized functions for character type identification, Excel requires creative use of nested Excel functions to isolate specific elements, such as the very first letter within a mixed alphanumeric sequence. This need frequently arises when dealing with standardized identifiers, such as employee IDs, product codes, or financial transaction labels, where the text component holds crucial classification information.

The core difficulty lies in determining where the numerical sequence ends and the alphabetical sequence begins, or vice versa. Since Excel lacks a straightforward built-in function like REGEX that can easily pattern-match for the first non-numeric character, we must construct a robust formula that iterates through every character of the string, testing each one to see if it is text. This intricate process involves leveraging array logic to test the entire string simultaneously, making the solution both powerful and efficient for large datasets.

In this comprehensive guide, we will explore two highly effective, albeit complex, Excel functions designed for this purpose. The first formula identifies and returns the numerical position of the first letter, while the second builds upon this knowledge to extract the actual character value. Understanding these formulas requires a solid grasp of how Excel handles array operations and error checking.

Core Strategy: Using Error Handling and Array Logic to Identify Text

The fundamental strategy used in these advanced formulas relies on forcing Excel to evaluate whether each character within the target string is a number. By attempting to convert every single character to a numeric value, we can create a powerful TRUE/FALSE array that flags where non-numeric characters (i.e., letters or symbols) reside. If the character is successfully converted to a number, the process is smooth; if the character is a letter, the conversion attempt fails, generating an error.

Specifically, this involves nesting three critical Excel functions: MID, VALUE, and ISERROR. The MID function breaks the string into individual characters. The VALUE function attempts the numeric conversion. Finally, ISERROR converts the resulting errors into TRUE values, signifying that the character in that position is indeed a non-numeric character (a letter).

Once this logical array of TRUE/FALSE values is constructed, we can use the MATCH function to search for the first occurrence of TRUE. Because MATCH returns the relative position of the match within the array, this effectively gives us the position index of the first letter found in the original string. This technique is both elegant and required, demonstrating the power of iterative array

processing within Excel's calculation engine.

Solution 1: Returning the Position of the First Letter

The first practical goal is to determine the exact index or position of the first alphabetical character within the target cell. This position is vital, as it serves as the key input for extraction functions like MID, which requires a starting position argument. If we have a mixed string like **X9903Y**, the desired result for this formula would be **1**, indicating that the letter 'X' is the first character.

The following comprehensive formula achieves this by generating an array of positions equal to the length of the string and testing each position for non-numeric content.

Formula 1: Return Position of First Letter

=MATCH(TRUE,ISERROR(VALUE(MID(A2,ROW(INDIRECT("1:"&LEN(A2))),1))),0)

This complex formula returns the numerical position (index) of the very first letter found in the string located in cell A2. For instance, if the string contained in A2 is **A0095B**, the formula will accurately return the value **1**, as the letter 'A' occupies the initial position. If the string were **99B001**, the result would be **3**.

Detailed Breakdown of Formula 1 Components

To truly master this technique, it is essential to dissect how the various functions interlock to produce the final result. The formula works from the inside out, beginning with string decomposition and array generation.

1. **Generating Character Positions (ROW, LEN, INDIRECT):** The combination of LEN and INDIRECT creates a reference that generates a sequence of numbers from 1 up to the total length of the string. For a string **A0095B** (length 6), this generates the array `{1; 2; 3; 4; 5; 6}`. This array is then fed into the MID function as the starting position argument.

2. **Extracting Individual Characters (MID):** The MID function uses the array of positions generated above to extract every single character of the string, one by one, resulting in a new array. For **A0095B**, the result is the array `{"A"; "0"; "0"; "9"; "5"; "B"}`.

3. **Forcing Numeric Conversion and Error Generation (VALUE):** The VALUE function is applied to this character array. It successfully converts numeric strings like "0" and "9" into actual numeric values, but it fails on text characters like "A" and "B," returning the `#VALUE!` error. The resulting array looks like: `{#VALUE!; 0; 0; 9; 5; #VALUE!}`.

4. **Identifying Non-Numeric Characters (ISERROR):** The ISERROR function checks the array for any error results. Wherever an error exists (which signifies a letter), it returns TRUE. Where the conversion was successful (numbers), it returns FALSE. The array is transformed into: `{TRUE; FALSE; FALSE; FALSE; FALSE; TRUE}`.

5. **Finding the First TRUE (MATCH):** Finally, the MATCH function searches this final TRUE/FALSE array for the first instance of TRUE, using the exact match type (0). Since TRUE is located at the first position in our example array, MATCH returns the index **1**.

Practical Application: Finding the Position (Example 1)

Let us demonstrate the application of Formula 1 using a common scenario involving a list of employee ID strings that contain a mix of letters and numbers. We aim to populate Column B with the starting position of the first text character for each ID listed in Column A.

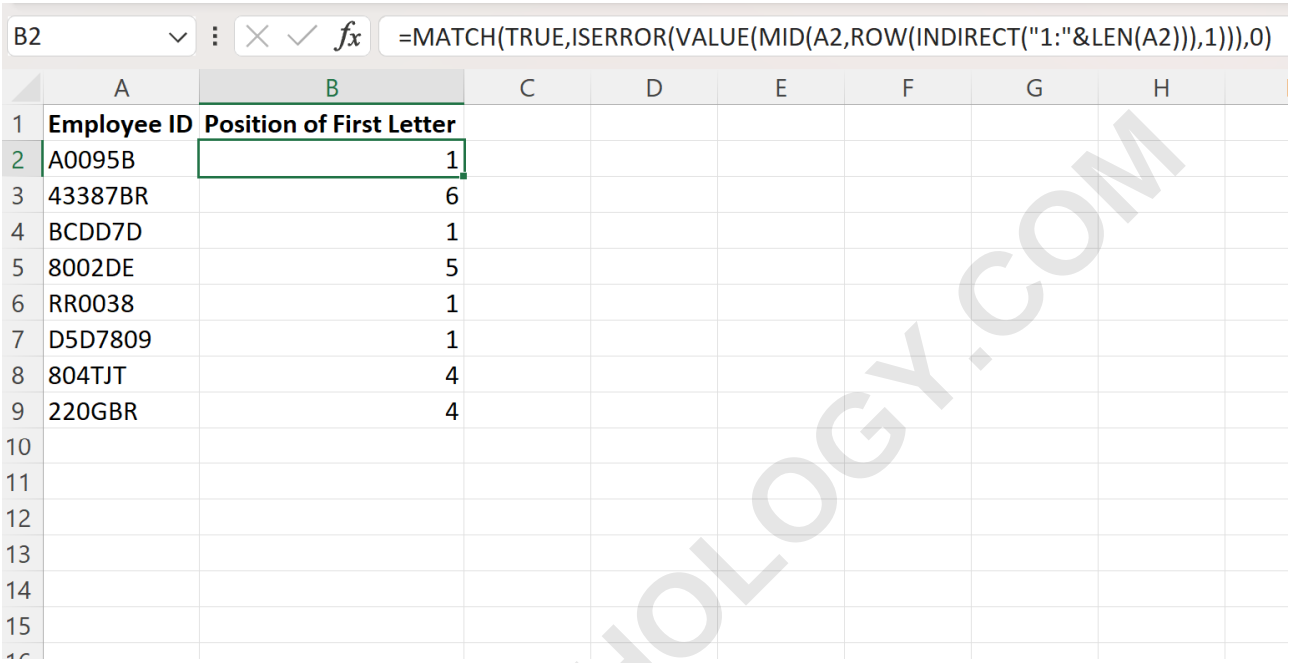
Suppose we have the following dataset where employee IDs are listed in Column A:

	A	B	C	D	E
1	**Employee ID**				
2	A0095B				
3	43387BR				
4	BCDD7D				
5	8002DE				
6	RR0038				
7	D5D7809				
8	804TJT				
9	220GBR				
10					
11					
12					
13					
14					
15					
16					

We initiate the process by inputting the first formula into cell **B2** to calculate the position for the string in A2:

=MATCH(TRUE,ISERROR(VALUE(MID(A2,ROW(INDIRECT("1:"&LEN(A2))),1))),0)

After entering the formula into B2, we utilize the fill handle--the small square at the bottom right of the cell--to efficiently click and drag this formula down to apply it across all remaining cells in Column B. This action calculates the position for every corresponding employee ID string in Column A.



	A	B	C	D	E	F	G	H
1	**Employee ID**	**Position of First Letter**						
2	A0095B	1						
3	43387BR	6						
4	BCDD7D	1						
5	8002DE	5						
6	RR0038	1						
7	D5D7809	1						
8	804TJT	4						
9	220GBR	4						
10								
11								
12								
13								
14								
15								

As illustrated in the resulting table, Column B successfully returns the position of the first letter for each respective string in Column A, confirming the efficacy of the array-based approach. The results provide precise indices for subsequent data manipulation tasks.

For the string **A0095B**, the first letter 'A' is found at position **1**.
For the string **43387BR**, the first letter 'B' is found significantly later, at position **6**.
For the string **BCDD7D**, the first letter 'B' is immediately found at position **1**.

This pattern continues, ensuring that complex, mixed strings are accurately parsed to find the start of the alphabetical sequence.

Solution 2: Extracting the Value of the First Letter

While knowing the position is useful, the ultimate objective is often to extract the actual alphabetical character itself. Fortunately, the robust logic developed in Formula 1 provides the exact index needed to execute the extraction using the versatile MID function. By nesting Formula 1 as the starting position argument within MID, we can create a single, powerful formula that directly returns the desired character.

The structure of this second solution is inherently dependent on the success of the first. We use the output of the MATCH function (the position) to tell MID exactly where to begin extracting.

Formula 2: Return Value of First Letter

=MID(A2,MATCH(TRUE,ISERROR(VALUE(MID(A2,ROW(INDIRECT("1:"&LEN(A2))),1))),0),1)

This sophisticated formula returns the actual character value of the first letter found within the string in cell A2. For example, if the string is **A0095B**, the embedded positioning mechanism determines the position is 1, and the outermost MID function extracts the character at position 1, returning **A**. If the string is **43387BR**, the position is calculated as 6, and MID extracts the character **B**.

Detailed Breakdown of Formula 2: Integrating Extraction

Formula 2 is fundamentally Formula 1 wrapped inside the MID function. Understanding its arguments clarifies its operation:

Text Argument: The first argument is simply the reference to the original string, **A2**, from which the extraction will occur.

Start_num Argument (The Engine): The second argument, where the position is typically specified, is replaced entirely by Formula 1: `MATCH(TRUE,ISERROR(VALUE(MID(A2,ROW(INDIRECT("1:"&LEN(A2))),1))),0)`. This entire nested calculation determines the precise starting index of the first letter. If this internal calculation returns **6** (as in the case of **43387BR**), then 6 becomes the starting position for the extraction.

Num_chars Argument: The final argument specifies how many characters to extract starting from the determined position. Since we only want the single first letter, this value is set to **1**.

This integrated approach allows the entire string analysis and extraction process to be conducted within a single cell, providing a clean and maintainable solution for data preprocessing and cleaning tasks. It is important to remember that in older versions of Excel (pre-Office 365 dynamic array formulas), these formulas often require confirmation by pressing Ctrl+Shift+Enter to execute them as an array formula, surrounding the formula with curly braces.

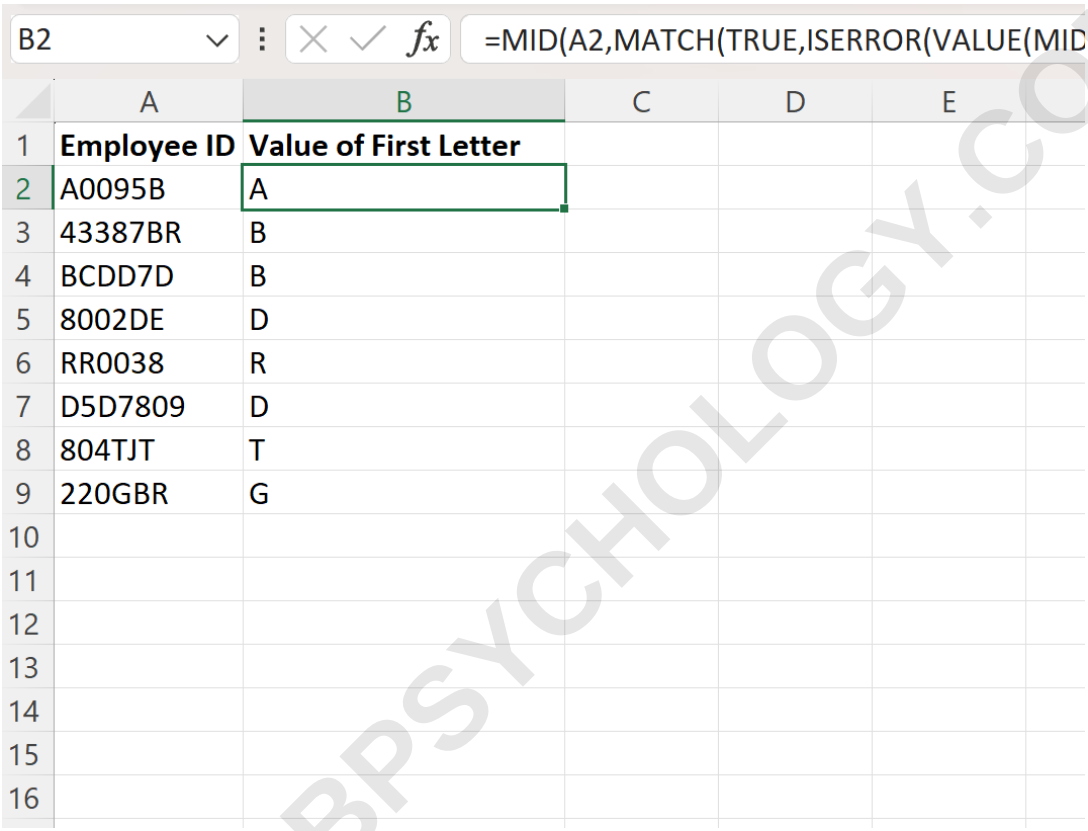
Practical Application: Extracting the Value (Example 2)

We now apply Formula 2 to the same dataset of employee ID strings to extract the actual alphabetical component that begins the text sequence. This is typically the most desired output when classifying or filtering data based on string composition.

We enter the full extraction formula into cell **B2**:

=MID(A2,MATCH(TRUE,ISERROR(VALUE(MID(A2,ROW(INDIRECT("1:"&LEN(A2))),1))),0),1)

Following successful entry, we again use the click-and-drag method to cascade the formula down Column B, populating the column with the extracted first letters for all corresponding employee IDs in Column A. This action efficiently processes the entire list.



	A	B	C	D	E
1	**Employee ID**	**Value of First Letter**			
2	A0095B	A			
3	43387BR	B			
4	BCDD7D	B			
5	8002DE	D			
6	RR0038	R			
7	D5D7809	D			
8	804TJT	T			
9	220GBR	G			
10					
11					
12					
13					
14					
15					
16					

The final results in Column B clearly demonstrate the extraction of the initial letter. This provides a clean dataset ready for further analysis, such as categorizing employees based on the starting letter of their code.

For the ID **A0095B**, the extracted value is **A**.
For the ID **43387BR**, the extraction correctly returns **B**, skipping the leading numeric sequence.
For the ID **BCDD7D**, the first letter extracted is **B**.

This confirms that the formula successfully isolates the first text character, regardless of preceding numeric content.

Conclusion: Mastering Complex String Manipulation

Mastering complex string manipulation in Excel, particularly when dealing with mixed alphanumeric data, relies heavily on understanding the concept of array processing and error handling. By constructing these powerful nested formulas using ISERROR, VALUE, and MATCH, Excel users can overcome the limitations of standard text functions and accurately isolate the first letter in any given string.

These solutions provide robust methods for data cleaning and preparation, ensuring that crucial text components embedded within mixed data structures can be quickly identified and utilized for reporting and analysis. Whether you need the position index or the extracted character value, these two formulas serve as fundamental tools in the advanced Excel user's toolkit for handling real-world data ambiguities.