Learn How to Split Text Using Multiple Delimiters in Google Sheets

Authored by

Mohammed looti

November 14, 2025

The efficient manipulation of raw text data stands as a critical requirement in modern data analysis and management. For users of **Google Sheets**, the platform provides robust utilities specifically designed to handle this task. Chief among these is the **SPLIT function**, a fundamental tool engineered to decompose a single string of text into separate columns. This process relies entirely on identifying a specific character or sequence of characters--known as a **delimiter**--which marks the division points within the data. Utilizing **SPLIT** is the standard method for transforming consistently structured data, such as records separated only by commas or pipes, into a neatly organized tabular format.

However, the complexity inherent in real-world datasets often quickly outpaces the simple requirements of a singular **delimiter**. Data frequently arrives in an inconsistent state, where text elements might be separated interchangeably by underscores, spaces, commas, or semicolons--all serving the same logical purpose of separation. When confronted with this assortment of diverse separators, the native **SPLIT function** proves inadequate. It is fundamentally designed to accept either a single character or a fixed sequence of characters as its dividing criteria, rendering it ineffective when the separation logic demands the matching of multiple, alternative patterns simultaneously.

This comprehensive guide details an advanced and essential technique that empowers the **SPLIT function** to successfully process a multiplicity of **delimiters** within **Google Sheets**. The strategy involves strategically nesting the **SPLIT function** with the highly versatile **REGEXREPLACE function**. By doing so, we introduce a crucial preprocessing step that unifies all disparate separators into a single, predictable character before the final division occurs. This method dramatically enhances the flexibility, precision, and robustness of your data parsing operations, significantly streamlining complex **Google Sheets** workflows.

The Challenge of Inconsistent Delimiters in Text Parsing

The primary goal of text parsing is to efficiently extract discrete pieces of information from a larger, aggregated string of characters. For straightforward cases, the **SPLIT function** is perfectly suitable, requiring only the source text and the singular **delimiter** that designates the points of separation. For example, separating "John Doe" into two distinct columns is effortless when using the space character (" ") as the sole separator.

However, real-world data is seldom that clean. Data compilation often results in inherent inconsistencies, requiring more sophisticated handling. Consider a scenario where a list of user names has been aggregated from various sources, leading to entries formatted as "Jane_Doe", "Richard,Roe", and "Sam;Smith". In this problematic situation, we are presented not with a single separator, but with three distinct characters--the underscore, the comma, and the semicolon--all sharing the identical logical purpose of dividing the first name from the last name. The conventional

SPLIT function lacks the native capability to process this complex "OR" logic; it can only interpret a single string as the dividing criterion, or alternatively, treat every character in the delimiter string individually, which frequently results in unintended fragmentation and inaccurate data splits.

To successfully overcome this significant data hurdle, it is imperative that we introduce a preliminary normalization step. Before the text can be divided reliably, all variant **delimiters** must be converted into a single, standardized character that the **SPLIT function** is capable of processing. This essential preparatory standardization is where the advanced pattern-matching capabilities of the **REGEXREPLACE function** become indispensable, allowing us to define and replace complex separation patterns using the power of **regular expressions**.

The Two-Phase Strategy: REGEXREPLACE for Normalization

The robust solution for managing multiple **delimiters** relies on a highly efficient two-phase strategy, which is executed seamlessly within a single, nested formula in **Google Sheets**. The critical first phase utilizes the **REGEXREPLACE function** to standardize the input text by locating all diverse separators and converting them into one uniform **delimiter**. The second phase takes the output of this newly standardized text and passes it directly into the outer **SPLIT function** for the final, clean division into columns.

The **REGEXREPLACE function** is specifically designed to find and replace text based on sophisticated search patterns known as **regular expressions**. Unlike simpler functions like `SUBSTITUTE`, `REGEXREPLACE` allows us to define a highly flexible pattern that essentially translates to: "match and replace character A OR character B OR character C." This powerful logical disjunction is achieved using the vertical bar (|) operator within the expression, providing a highly efficient mechanism for bulk replacement of varied characters.

By employing **REGEXREPLACE**, we achieve the necessary preprocessing step known as normalization. For instance, if our goal is to target commas, underscores, and semicolons, we can instruct the function to replace any instance of these three characters with a single space. Once this normalization is complete, the resulting string is uniformly delimited by spaces, making it perfectly structured for the final step: reliable division by the **SPLIT function**. This synergy ensures that even the most inconsistent source data can be managed and transformed using a reliable and repeatable formula structure.

Formula Syntax: Mastering the Nested Structure

A thorough understanding of the nested formula's architecture is crucial for successfully implementing this advanced parsing technique. The design relies on embedding the `REGEXREPLACE` function within the `SPLIT` function, which ensures that the text is fully standardized before the division process begins. This logical structure guarantees a clean,

sequential processing flow, which is absolutely critical for effectively handling multiple **delimiters**.

The standard syntax for this robust data cleaning solution is structured as follows:

=SPLIT(REGEXREPLACE(text_to_split, "regex_pattern_of_delimiters", "single_delimiter"), "single_delimiter")

Let us dissect the roles of the arguments within this powerful formula. The innermost function, **REGEXREPLACE**, requires three essential parameters: first, the text string itself (which typically references a cell, such as `A2`); second, the **regular expression** pattern that matches all desired variant separators (e.g., `"_,|;"`); and third, the replacement string, which is most often defined as a single space (`" "`). This initial step is exclusively responsible for unifying all variant separation characters into a consistent standard.

The output generated by this standardization step--the text string with all variant separators replaced by a single, standard character--is then utilized as the primary argument for the outer **SPLIT function**. Because all original **delimiters** have already been converted to the standard character (e.g., a space), **SPLIT** can then confidently use that standard character (e.g., `" "`) as its defined **delimiter**. This dual-function methodology guarantees the accurate and reliable separation of data elements into adjacent columns, regardless of any initial inconsistencies found in the source text.

Practical Implementation: Handling Messy Source Data

To fully grasp the practical efficiency of this combined formula, it is helpful to visualize a common data aggregation scenario. Imagine you have collected a large list of names, but due to the merger of data from several different entry systems, the first and last names are separated using various inconsistent methods. This mixed input format is a frequent occurrence when importing raw logs or consolidating various spreadsheets.

Suppose your **Google Sheets** column contains the following example of mixed data:

	A	B	C	D
1	**Name**			
2	Andy Bernard			
3	Bob_Erickson			
4	Chad_Davis			
5	Dean,Anderson			
6	Eric Wilbor			
7	Frank;Johnson			
8	George,Carl			
9	Henry_Miller			
10	Isaac King			
11	John;Freeman			
12				
13				
14				
15				
16				
17				

A quick inspection of this challenging dataset reveals significant variation in the separators employed. These diverse **delimiters** include standard spaces ( ), underscores (_), commas (**,**), and semi-colons (**;**). If we were to attempt to use a simple **SPLIT function** based on any single one of these characters, the function would fail to accurately process the remaining rows in the list. This inconsistency mandates a more robust and flexible solution.

Our objective is to execute a reliable split, extracting both the first name and the last name into two separate columns, irrespective of which specific **delimiter** was used in the original string. This challenge represents the ideal use case for the powerful combination of **REGEXREPLACE** and **SPLIT**, which together provide a clean, efficient, and scalable methodology for comprehensive data normalization and extraction.

Step-by-Step Guide for Implementation

Assuming the column containing the messy, inconsistent names is located in Column A of your spreadsheet, starting from cell A2, we will implement the combined formula in cell B2. This formula will first standardize the varied separators and then execute the split operation, placing the extracted first name in Column B and the last name in Column C.

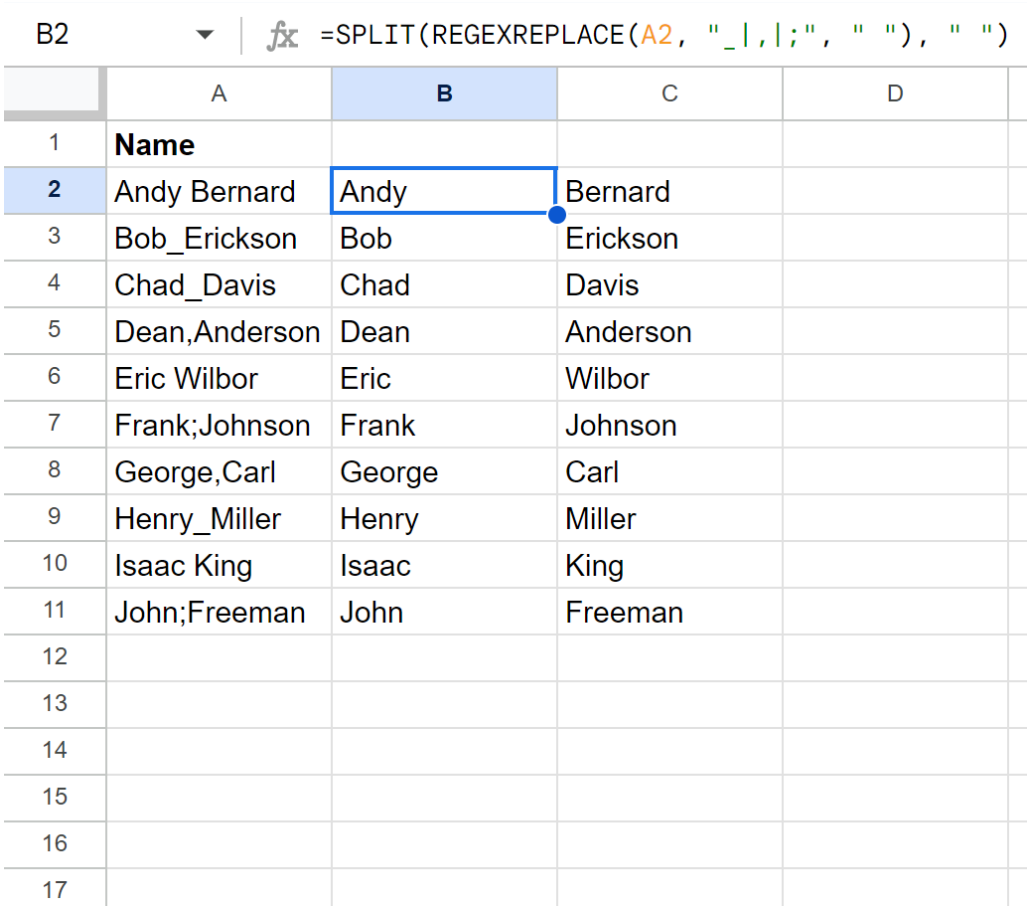
Select cell **B2**, which is designated as the starting point where the parsed output will appear.
Input the following formula precisely into cell **B2**. This specific formula is configured to target the

underscore, comma, and semi-colon as alternative delimiters, convert them all into spaces, and subsequently split the resulting string using that space:

=SPLIT(REGEXREPLACE(A2, "_|,|;", " "), " ")

Once the formula is entered and confirmed by pressing Enter, **Google Sheets** executes the precise two-step process on the content of cell A2. The first part of the split text (the first name) will automatically populate B2, and the second part (the last name) will "spill" into the adjacent cell, C2. This automatic expansion across adjacent cells is a standard and highly convenient feature of the array output generated by the **SPLIT function**.

To apply this powerful logic across all remaining data rows, click back onto cell **B2**. Locate the small fill handle (the square dot) positioned at the bottom-right corner of the cell boundary. Click and drag this handle downward across all corresponding rows in your dataset. This action intelligently copies the formula, ensuring that the cell reference is correctly updated (from `A2` to `A3`, `A4`, and so on) for each row, thereby guaranteeing consistent and accurate parsing across the entire list.



	A	B	C	D
1	**Name**			
2	Andy Bernard	Andy	Bernard	
3	Bob_Erickson	Bob	Erickson	
4	Chad_Davis	Chad	Davis	
5	Dean,Anderson	Dean	Anderson	
6	Eric Wilbor	Eric	Wilbor	
7	Frank;Johnson	Frank	Johnson	
8	George,Carl	George	Carl	
9	Henry_Miller	Henry	Miller	
10	Isaac King	Isaac	King	
11	John;Freeman	John	Freeman	
12				
13				
14				
15				
16				
17				

As clearly demonstrated in the resulting table, the combined formula successfully and reliably splits every name from Column A into two clean, distinct columns. The accuracy remains perfect regardless of whether the original separation character was a space, an underscore, a comma, or a semi-colon, thereby proving the formula's effectiveness in achieving clean data extraction from highly varied input formats.

Customizing the Regular Expression for New Delimiters

The true power and longevity of the `SPLIT(REGEXREPLACE(...))` pattern reside in its high degree of customization and adaptability. Users are not confined to utilizing only the specific delimiters demonstrated in the introductory example. You possess the complete ability to modify the **regular expression** pattern string found within the **REGEXREPLACE function** to accommodate any specific set of separators relevant to your unique dataset.

If, for example, your data utilizes hyphens (`-`) and forward slashes (`/`) in addition to, or instead of, the previously defined separators, you would simply adjust the pattern string to include these new characters. To accurately split text based on hyphens, forward slashes, and spaces, the **regular expression** pattern would be written precisely as `"-|/| "`. You can easily aggregate any required number of **delimiters** by chaining them together using the logical `|` operator within the quoted string argument.

It is important to remember that when constructing **regular expressions**, certain characters (known as metacharacters) hold special functional meaning within the regex engine, such as `.`, `*`, `+`, `(`, `)`, and `[`. If your intention is to match one of these special characters literally within your text, you must precede it with a backslash () to "escape" its special meaning. This remarkable adaptability makes the combined formula an indispensable and flexible technique for data cleaning and preparation in **Google Sheets**, granting immense control over the most complex text manipulation challenges.

Conclusion

The imperative to parse text data accurately is a constant requirement in all forms of data analysis, and the presence of inconsistent, multiple **delimiters** frequently presents a significant barrier to clean processing. However, this guide has clearly demonstrated that by intelligently combining **Google Sheets**' foundational **SPLIT function** with the sophisticated pattern-matching capabilities of the **REGEXREPLACE function**, a robust, reliable, and scalable solution can be achieved.

This powerful technique relies on a crucial and logical two-step methodology: first, leveraging **REGEXREPLACE** and **regular expressions** to standardize all varied separators into a single, uniform character; and second, utilizing the standard **SPLIT function** to efficiently divide the resulting normalized text. This sequential approach guarantees a consistent and immaculately

clean output, regardless of how disorganized the initial data input may have been.

Mastering the nested `SPLIT(REGEXREPLACE(...))` combination provides a significant and powerful enhancement to your entire data processing toolkit within **Google Sheets**. Whether your regular tasks involve cleaning messy imported data, standardizing user inputs from forms, or extracting specific fields from complex, semi-structured text strings, this method empowers you to handle challenging text manipulation requirements with unparalleled precision and operational efficiency.

Additional Resources for Advanced Text Manipulation

For those committed to further developing their expertise in **Google Sheets** text manipulation and the construction of advanced nested formulas, the following official resources and tutorials offer valuable, in-depth documentation:

Google Sheets SPLIT Function Documentation
Google Sheets REGEXREPLACE Function Documentation
Introduction to Regular Expressions (Wikipedia)