How to Easily Remove Special Characters in Google Sheets

Authored by

stats writer

November 30, 2025

When working with large datasets in Google Sheets, one of the most common challenges faced by analysts and data entry specialists is the presence of extraneous or non-standard characters. These unexpected elements, often referred to as special characters--such as punctuation marks, symbols, or hidden control codes--can severely disrupt calculations, filtering, and database imports. Effective data cleaning is crucial for maintaining data integrity, ensuring that reports are accurate, and optimizing the flow of information for subsequent analysis or integration with other systems. Ignoring these characters can lead to errors in formulas, inaccurate aggregations, and formatting issues that undermine the credibility of the entire dataset.

Fortunately, Google Sheets provides powerful, built-in functions designed specifically to handle complex text manipulation and character removal. While simpler tasks might be handled using the `SUBSTITUTE` function to replace a known, singular character with an empty string, dealing with a vast, unpredictable array of symbols requires a more sophisticated approach. This is where the true power of advanced functions like `REGEXREPLACE` comes into play. By leveraging regular expressions, users gain precise control over which characters are kept, which are removed, and how patterns within the data are identified and manipulated, thereby streamlining the process of data sanitization across thousands of rows instantly.

This guide explores the most effective and efficient methods for eliminating unwanted symbols from your spreadsheets. We will focus primarily on using the versatile REGEXREPLACE function, which utilizes powerful pattern matching to target and scrub undesirable elements, whether you need to isolate only letters, retain alphanumeric values, or remove a defined set of specific symbols. Furthermore, we will touch upon alternative techniques, including using the basic 'Find and Replace' feature for manually addressing specific, known instances of clutter. Mastering these techniques ensures your data remains clean, standardized, and ready for accurate processing and visualization, transforming raw, messy input into reliable, structured information.

The following formulas utilize the power of regular expressions to expertly remove special characters in Google Sheets:

The Power of Regular Expressions in Google Sheets

The key to efficient and comprehensive character removal in any modern spreadsheet application, including Google Sheets, lies in understanding and utilizing regular expressions, commonly known as Regex. Regex provides a specialized syntax for searching and manipulating strings based on sophisticated patterns rather than fixed text strings. Instead of writing a formula that substitutes one specific character at a time, a single Regex pattern can define an entire class of characters--such as all non-alphanumeric symbols, all whitespace, or all punctuation--allowing for bulk removal that is both faster and significantly more robust than manual methods or simple functions like `SUBSTITUTE`.

The REGEXREPLACE function is specifically designed to apply these patterns. Its syntax is straightforward: `REGEXREPLACE(text, regular_expression, replacement)`. The function takes the original text, compares it against the defined regular expression pattern, and wherever a match is found, it replaces that match with the specified replacement text. For the purpose of removing characters, the replacement text is typically an empty string (`""`), effectively deleting the matched pattern from the cell. This function is indispensable when dealing with data imported from disparate sources, customer feedback forms, or web scraping results, where messy formatting and unexpected symbols are rampant.

When constructing Regex patterns for character removal, it is essential to focus on character classes and negation. Instead of trying to list every special character you want to remove (which is often impossible), it is far more efficient to define what you want to keep. The caret symbol (`^`) inside square brackets () acts as a negation operator. For example, the pattern means "match any character that is **NOT** an uppercase letter from A to Z." By defining the acceptable characters, we automatically target and eliminate everything else, including all varieties of special characters, punctuation, and whitespace, providing a clean slate for subsequent analysis. The three methods detailed below illustrate how to leverage this negation principle for different data cleaning scenarios.

Method 1: Comprehensive Data Isolation (Remove Everything Except Letters)

This method is utilized when the objective is to retain only alphabetical characters--a common requirement when standardizing names, cleaning up location data, or preparing textual input for linguistic analysis where numbers and symbols are considered noise. The approach uses a single powerful regular expression to define the acceptable range of letters, thereby discarding everything that falls outside that strict definition. This ensures that even accidental spaces, control characters, or non-visible symbols are scrubbed from the cell content, leaving behind only the pure textual component of the data.

The core of this technique lies in the pattern `+`. Let's break down this powerful sequence: the square brackets define a character set. The caret `^` at the beginning of the set means "NOT" any of the characters listed within the brackets. `A-Z` covers all uppercase English letters, and `a-z` covers all lowercase English letters. Therefore, matches any character that is not a standard English letter. Finally, the plus sign `+` is a quantifier that means "one or more" occurrences of the preceding pattern. This ensures that consecutive special characters are matched and removed in a single operation, improving efficiency. This formula is invaluable for fields where numeric input or symbols are errors, such as person names or state abbreviations.

To implement this, you simply reference the cell containing the raw data (e.g., A1) and set the replacement value to an empty string. The resulting formula is highly efficient and scalable, cleaning large columns of text data instantly. This method is often the first step in normalizing

textual data before applying further transformations, such as converting text to proper case or standardizing capitalization, ensuring the consistency required for robust data management practices.

=REGEXREPLACE(A1,"+","")

Method 2: Alphanumeric Preservation (Remove Everything Except Letters & Numbers)

In many business and analytical contexts, data fields inherently contain both textual and numeric information, such as product codes, addresses, inventory SKUs, or specialized reference numbers. In these scenarios, the goal is not to remove all non-letters, but specifically to eliminate only the extraneous special characters and punctuation marks that interfere with data processing, while preserving both the letters and the digits. This method requires a slight adjustment to the Regex pattern used in Method 1, expanding the accepted character set to include numbers.

The tailored pattern for this objective is . By adding `0-9` within the negated character class, we instruct the REGEXREPLACE function to exclude digits (0 through 9), alongside both uppercase and lowercase letters, from the removal process. consequently, any character found in the referenced cell that is not a letter or a number will be successfully matched and replaced with an empty string. This is particularly effective for standardizing serial numbers or unique identifiers that might inadvertently pick up dashes, slashes, or symbols during manual entry or data migration.

While the pattern is highly effective, users should be aware that it also removes whitespace. If spaces are necessary to separate words (e.g., in an address line where only punctuation is noise), the pattern must be modified to explicitly include the space character: . However, for maximum data standardization and when generating unique identifiers, removing all non-alphanumeric characters, including spaces, is often the preferred and cleaner approach, ensuring consistency across all data entries ready for import into a database system.

=REGEXREPLACE(A1, "", "")

Method 3: Targeted Removal (Remove Specific Special Characters)

Sometimes, the data cleaning task is highly specific: you only need to remove a small, known subset of special characters that are known to cause problems, while leaving all other characters, including numbers, letters, and even other symbols, intact. For example, perhaps you need to eliminate only dollar signs, percentage marks, and ampersands without affecting any other punctuation or formatting within the text. This scenario calls for a non-negated character set within the REGEXREPLACE function.

To achieve this precise targeting, the regular expression pattern should list the specific characters intended for removal inside the square brackets, without the initial negation caret (^). The example pattern used here is . Since the caret is absent, this pattern specifically matches any occurrence of an exclamation mark, a dollar sign, or a percentage sign. When this pattern is passed to `REGEXREPLACE`, only these three characters are targeted and replaced with the empty string, leaving all other elements untouched. This approach offers fine-grained control for addressing localized data corruption or removing symbols that might have been mistakenly included during data capture.

Customizing this method is straightforward. If you needed to remove commas, semicolons, and parentheses, you would simply update the pattern to . Crucially, when defining character sets in Regex, certain characters must be escaped if they have special meaning in the Regex syntax (e.g., `.` or `*`). However, inside square brackets, most common punctuation marks (like those used in the example) lose their special meaning and can be included directly. This makes the targeted removal method highly flexible and easy to adapt to unique data requirements, providing a clean method for specific text standardization.

=REGEXREPLACE(A1, "", "")

The following examples show how to use each formula in practice, demonstrating the immediate impact of the regular expressions on various data types.

Example 1: Isolating Textual Data

This example demonstrates the application of Method 1, focusing on extracting pure alphabetical text from a cell that contains a mix of letters, numbers, and symbols. Imagine a scenario where imported client names have been polluted with accidental project codes and special characters from a faulty integration. The objective is to retrieve only the client's name, discarding all other non-letter characters that prevent accurate sorting and categorization.

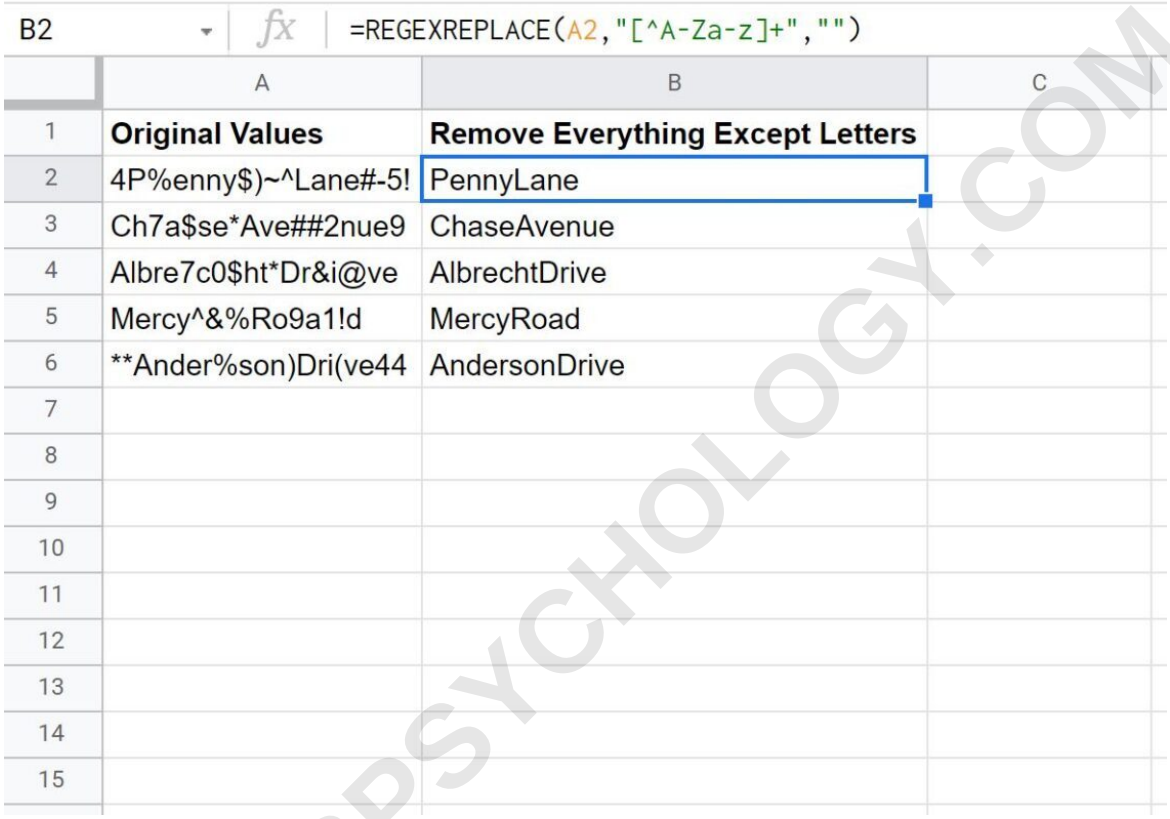
We use the formula employing the powerful negation pattern +. This pattern acts as a precise filter, identifying and marking for replacement every character that is not an English alphabet letter, regardless of case. By applying this function to the source cell, all numbers, punctuation marks, and whitespace are instantly eliminated, leaving a clean, standardized text string suitable for entry into a contact database or CRM system. This process significantly contributes to the overall integrity of the contact list, ensuring that only valid textual data is retained.

The resulting output clearly demonstrates the effectiveness of defining what to keep, rather than listing what to remove. The entire string of non-letter elements is condensed into nothing, proving the efficiency of the REGEXREPLACE function for high-volume data cleaning tasks. Pay close

attention to how complex, interwoven characters are completely stripped away, illustrating the function's utility beyond simple, single-character substitution.

=REGEXREPLACE(A2,"+","")

The following screenshot shows how to use this formula in practice:



Reviewing the results, notice how every single special character and numeric digit has been successfully removed from the original cell values. This powerful isolation technique ensures that only the intended alphabetical content remains. This level of purity is often non-negotiable when preparing data for specialized applications that cannot tolerate non-textual inputs.

The remaining values consist solely of the letters extracted from each cell, merged together without any intervening spaces. If spaces are desired, the regular expression pattern must be adjusted to `+` (with a space included inside the brackets). However, for foundational data cleaning, combining all letters is often advantageous before applying specific formatting rules.

Example 2: Cleaning Alphanumeric Identifiers

This second example applies Method 2, which is critical for cleaning fields that are expected to

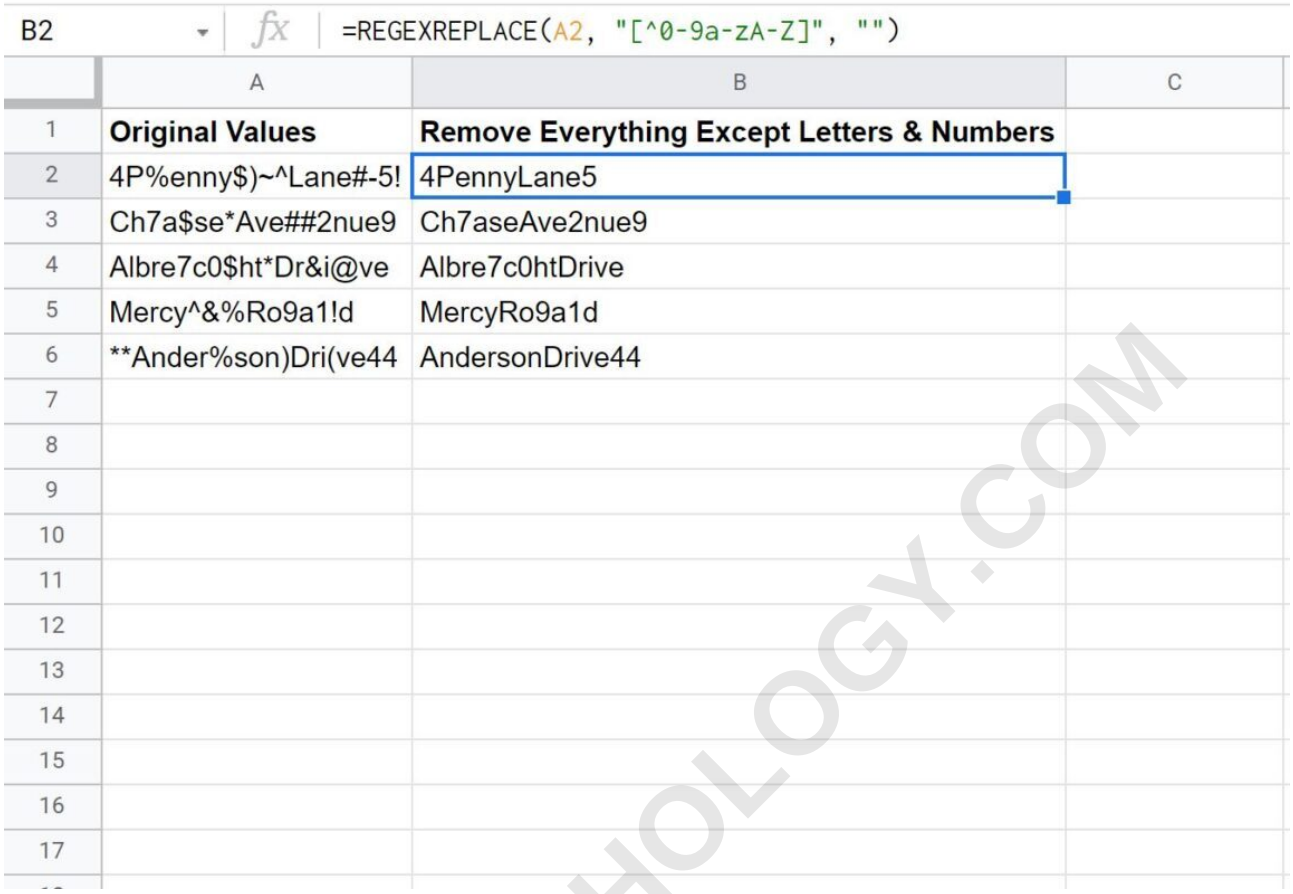
contain a mix of letters and numbers, such as part numbers, invoice IDs, or secure key codes. If these identifiers are compromised by errant dashes, semicolons, or currency symbols, they become unusable for matching, inventory tracking, or database lookups. The goal here is to retain the numerical and alphabetical sequence while eliminating all other noise.

We implement the formula utilizing the pattern . This expanded negation instructs the REGEXREPLACE function to preserve the full range of letters (A-Z, a-z) and all digits (0-9). Consequently, all symbols, punctuation, and non-alphanumeric characters are precisely targeted and removed. This method is superior to simple Find and Replace because it handles an infinite variety of special characters that might exist within the data, rather than requiring the user to identify and manually remove each one.

The transformation ensures that the resulting data is a standardized string of letters and numbers, which is the required format for virtually all database indexes and structured data tables. Observe how the original structure of the alphanumeric sequence is maintained, while the disruptive symbols are completely eliminated, confirming the data is now validated against the expected content type. This step is essential in any robust data cleaning workflow.

=REGEXREPLACE(A2, "", "")

The following screenshot shows how to use this formula in practice:



	A	B	C
1	**Original Values**	**Remove Everything Except Letters & Numbers**	
2	4P%enny$)~^Lane#-5!	4PennyLane5	
3	Ch7a$se*Ave##2nue9	Ch7aseAve2nue9	
4	Albre7c0$ht*Dr&i@ve	Albre7c0htDrive	
5	Mercy^&%Ro9a1!d	MercyRo9a1d	
6	**Ander%son)Dri(ve44	AndersonDrive44	
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			

Upon inspection of the outcome, it is clear that every special character has been successfully scrubbed from the initial values. Crucially, the numerical data embedded within the strings was preserved, demonstrating the precision of the alphanumeric filtering pattern. The final, cleaned string consists only of the letters and numbers, combined into a single, standardized identifier that is ready for automated processing or lookup operations within Google Sheets or external systems.

This confirms that by carefully defining the character class using the Regex notation, we can achieve highly targeted results. The ability to distinguish between noise (symbols) and necessary components (letters and numbers) makes the REGEXREPLACE function a fundamental tool in the arsenal of any serious spreadsheet user dealing with large volumes of unstructured or semi-structured data.

Example 3: Removing Specific Formatting Symbols

In contrast to the broad isolation techniques of the previous examples, this scenario focuses purely on targeted removal. Suppose a currency conversion process has left extraneous symbols like the dollar sign ($) and percentage sign (%) mixed with necessary punctuation and text. Using broad negation would eliminate necessary elements, but we only need to remove these specific

formatting characters. This is where Method 3 shines, offering surgical precision in text manipulation.

The formula implements the pattern . Since this pattern lacks the leading caret (^), it instructs the REGEXREPLACE function to match only these specific symbols: the exclamation point, the dollar sign, and the percentage sign. Every other character, including letters, numbers, spaces, and other punctuation (like commas or parentheses), is completely ignored and passed through to the final output unchanged. This is the ideal solution when the data structure is largely correct, but contaminated by a small, defined set of unwanted characters.

This targeted approach is essential for maintaining the contextual integrity of the data. For instance, if the original cell contained descriptive text with commas and periods, broad removal would destroy the grammatical structure. By using the targeted pattern, we only eliminate the disruptive symbols that interfere with numeric interpretation or visualization software, ensuring the remaining data remains structurally sound and readable while addressing the specific cleaning requirement.

We can use the following formula to remove specific special characters from a cell:

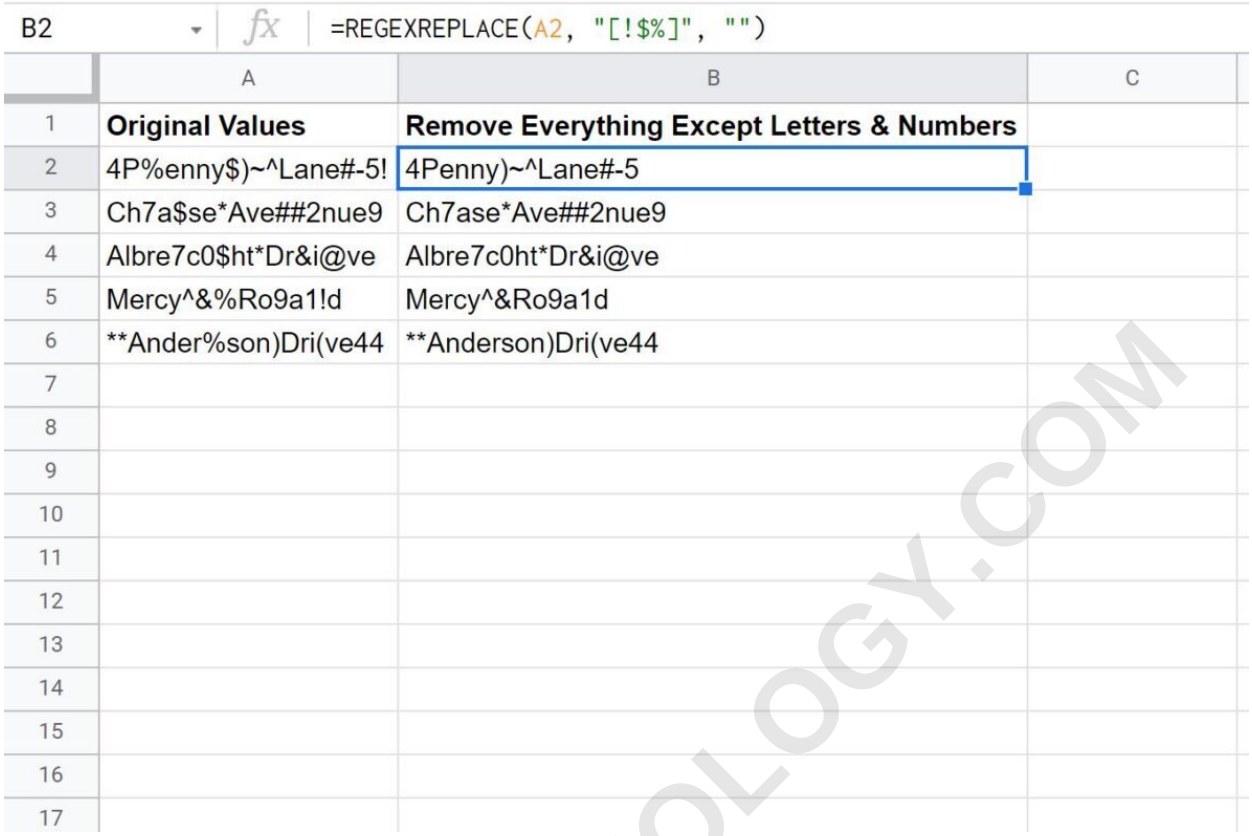
=REGEXREPLACE(A2, "", "")

This particular formula removes the following special characters from a cell, based on the definition within the square brackets:

The Exclamation Point (**!**)
The Dollar Sign (**$**)
The Percentage Sign (**%**)

Note that you can add as many specific special characters as you'd like between the brackets in the **REGEXREPLACE()** formula, provided you correctly escape any characters that hold special meaning in the regular expression syntax if they are outside of the brackets. For example, to also remove asterisks and hyphens, the pattern would become . This customization ensures that the data cleaning process perfectly aligns with the specific contaminants present in your dataset.

The following screenshot shows how to use this formula in practice:



Carefully observe the results: notice that every exclamation point (**!**), dollar sign (**$**), and percentage sign (**%**) has been successfully removed from the original cell content. However, other punctuation, such as the comma or the hyphen, which were not included in the targeted pattern, remain untouched. This highlights the controlled nature of targeted character removal using non-negated character classes within the regular expression framework.

All other values from the original cell, including all letters, numbers, and non-targeted punctuation, are preserved in the final output. This method proves invaluable when performing intermediate cleaning steps where full standardization is not yet desired, but specific disruptive symbols must be eliminated before moving on to the next stage of data transformation.

Alternative Methods: Simpler Techniques for Basic Cleaning

While `REGEXREPLACE` offers the ultimate flexibility and power for complex pattern matching, simpler text cleaning scenarios might be addressed using less complex, more direct methods. When you only need to remove a handful of known characters and do not require pattern matching, the `SUBSTITUTE` function or the built-in Find and Replace tool in Google Sheets can be sufficient and easier for novice users to implement.

The **`SUBSTITUTE` function** is ideal for replacing one specific character with another, or with

nothing. Its syntax is `SUBSTITUTE(text_to_search, search_for, replace_with, )`. To remove all instances of a hyphen, for example, you would use `=SUBSTITUTE(A1, "-", "")`. The limitation of `SUBSTITUTE` is that it can only handle one character replacement per function call. To remove multiple unique characters (e.g., both hyphens and asterisks), you would need to nest multiple `SUBSTITUTE` functions, which quickly becomes cumbersome and difficult to read. For example, to remove both characters: `=SUBSTITUTE(SUBSTITUTE(A1, "-", ""), "*", "")`.

The **Find and Replace** feature, accessible via the Edit menu, is the simplest non-formulaic way to achieve bulk character removal. By selecting 'Find and Replace' and entering the unwanted character in the 'Find' field and leaving the 'Replace with' field blank, you can quickly remove all occurrences across a selected range, the entire sheet, or the whole workbook. This method is fast and efficient for one-off cleaning tasks but is not dynamic; if new data is added, the manual process must be repeated. Furthermore, it only handles literal string matches and cannot leverage the pattern-matching capabilities of Regex, making it unsuitable for highly varied or complex data contamination.

Conclusion: Choosing the Right Tool for Data Integrity

Maintaining high data quality requires tools that are both powerful and adaptable. While basic functions like `SUBSTITUTE` and the manual Find and Replace option offer quick solutions for simple, known character removal, the ultimate efficiency in data cleaning in modern spreadsheet environments is achieved through the mastery of regular expressions. The `REGEXREPLACE` function provides the necessary flexibility to handle diverse and unpredictable data issues, allowing users to define complex rules for character retention or removal based on sophisticated patterns rather than relying on manual listing or character-by-character replacement.

For most data preparation tasks, whether isolating pure text, standardizing alphanumeric codes, or surgically removing targeted symbols, the Regex negation patterns provide the most robust and scalable solutions. By understanding the core principles of character classes and negation--defining what must be preserved ()--users can instantly sanitize hundreds or thousands of cells with a single, elegant formula. This not only saves significant time but also drastically reduces the risk of human error associated with manual cleaning processes.

Ultimately, incorporating these advanced text manipulation techniques into your standard workflow ensures that your datasets are always clean, consistent, and ready for advanced analysis or seamless integration with other database systems. Investing time in mastering the `REGEXREPLACE` function and its various patterns is perhaps the single most impactful step you can take toward becoming a highly proficient data analyst within the Google Sheets environment.