How to Easily Change Text Case in SAS: Uppercase, Lowercase, and Proper Case

Authored by

stats writer

December 1, 2025

Data standardization is a critical requirement in any statistical or analytical environment. When dealing with character variables, ensuring consistent capitalization is often necessary for accurate merging, comparison, and reporting. In the realm of SAS programming, developers and analysts frequently need to perform case conversion to normalize textual data, such as names, addresses, or product descriptions. Fortunately, SAS provides three powerful and intuitive built-in functions designed specifically for this task: **UPCASE**, **LOWCASE**, and **PROPCASE**.

These functions operate directly on a given string, transforming its characters according to specific capitalization rules. The **UPCASE** function systematically converts every alphabetic character within the input string to its corresponding uppercase form. Conversely, **LOWCASE** ensures that all letters are rendered in lowercase. Finally, **PROPCASE** applies a title-case format, capitalizing the first letter of every word while setting the remaining letters in lowercase, which is essential for human-readable output like proper nouns.

Understanding the application of these functions is fundamental for effective data cleaning in SAS. For instance, if the input text is "sas," applying **UPCASE** yields "SAS," **LOWCASE** keeps it "sas," and **PROPCASE** results in "Sas." Each function requires the original character variable as its sole argument and consistently returns the modified character string. Mastering these simple functions significantly enhances data quality and processing efficiency within the SAS environment.

Understanding SAS Character Functions

Character manipulation is a core aspect of statistical computing, and the reliability of your analysis often hinges on the quality of your textual data. SAS, or the Statistical Analysis System, offers a comprehensive library of functions specifically designed to handle character variables, ranging from simple trimming to complex pattern matching. The three case conversion functions—UPCASE, LOWCASE, and PROPCASE—belong to this essential toolset, ensuring that data records, regardless of their original input format, can be standardized into a uniform case structure for downstream processing.

When implementing these functions, it is crucial to understand that they operate element-wise. They take a source string as input and return a new, modified string. This means they are typically used within a SAS DATA step to create or modify variables. For example, if you have a variable named `customer_name` that contains mixed capitalization, applying one of these functions allows you to generate a new standardized variable, perhaps `standard_name`, ensuring every name follows the same casing rule.

The operational syntax for these conversions is remarkably straightforward, focusing on clarity and efficiency. The general structure involves assigning the result of the function call, which includes the variable or literal string to be modified, to a new or existing variable. Below, we outline the

fundamental syntax for each transformation before diving into detailed examples illustrating their application within a live dataset scenario.

Method 1: Convert String to Uppercase

new_string = UPCASE(old_string);

Method 2: Convert String to Lowercase

new_string = LOWCASE(old_string);

Method 3: Convert String to Proper Case

new_string = PROPCASE(old_string);

Utilizing the UPCASE Function for Standardization

The UPCASE function is arguably the most frequently used case function when preparing data for statistical modeling or database ingestion. By converting all text to **uppercase**, you eliminate discrepancies caused by inconsistent user input (e.g., 'IBM', 'ibm', or 'Ibm' all become 'IBM'). This standardization is crucial in scenarios where exact character matching is required, such as defining primary keys or performing lookups in external tables.

When applying **UPCASE**, the function iterates through the input variable byte-by-byte, identifying alphabetic characters and replacing them with their corresponding uppercase equivalents according to the system's character set (usually ASCII or EBCDIC). Non-alphabetic characters, such as numbers, symbols, and spaces, remain unaffected by the transformation process. This reliable behavior ensures data integrity while achieving the desired uniformity across the dataset.

A typical use case involves ensuring uniformity in category labels or identifiers. For example, if a survey dataset collects responses for state names, converting all entries to uppercase guarantees that 'California' and 'california' are correctly grouped as 'CALIFORNIA'. This simple step prevents classification errors and simplifies subsequent filtering and grouping operations using procedures like `PROC FREQ` or `PROC SQL` in SAS.

Utilizing the LOWCASE Function for Readability

While uppercase is often preferred for technical identifiers, converting strings to **lowercase** using the LOWCASE function enhances general readability, particularly when text mining or processing natural language data. The function operates identically to UPCASE, but converts all alphabetic

characters to their lowercase form. This is particularly useful in preprocessing steps for text analytics, where case sensitivity might unnecessarily inflate the vocabulary size or hinder accurate term frequency calculations.

The primary advantage of **LOWCASE** in analytical contexts is its contribution to text normalization. Before stemming or tokenization, converting all textual data to lowercase is standard practice to treat variations like "Apple" and "apple" as the same token. While **LOWCASE** might not be suitable for presentation-quality output where proper nouns are expected, it is an indispensable tool for data preparation where matching and feature extraction are the goals. It allows analysts to easily search for keywords without worrying about capitalization differences.

Consider a variable containing customer comments. If you want to count occurrences of specific keywords, failing to standardize the case means you would have to search for "Excellent," "excellent," and "EXCELLENT" separately. By applying the LOWCASE function upfront, the entire process is simplified, converting all comments into a uniform lowercase stream ready for deep analytical processing or simple keyword counting, thereby streamlining the overall workflow for case conversion.

Utilizing the PROPCASE Function for Presentation

The PROPCASE function, which stands for **Proper Case**, is designed for aesthetic and conventional data presentation. It applies title casing, meaning it capitalizes the first letter of every word within the string and converts all subsequent letters within that word to lowercase. This is the standard format used for names, titles, and geographical locations, maximizing readability and professional appearance in reports.

Defining what constitutes a "word" is key to understanding **PROPCASE**. In SAS, a word is typically defined as a sequence of characters separated by delimiters such as spaces, hyphens, or punctuation. The function intelligently identifies these separators to determine where a new word begins, ensuring the correct application of capitalization. For example, if the input is "san antonio spurs," PROPCASE correctly generates "San Antonio Spurs," adhering to standard writing conventions for proper nouns.

It is important to note that while **PROPCASE** is highly effective for presentation, it may sometimes require manual adjustment for complex names or specialized acronyms, as it strictly follows the "capitalize after delimiter" rule. However, for the vast majority of clean-up tasks involving names and addresses, the **PROPCASE function** offers an efficient, single-line solution for achieving presentation-ready proper case formatting, vastly superior to trying to implement complex conditional logic manually.

Data Preparation and Initial Setup in SAS

To demonstrate the practical application of these three functions, we will utilize a sample dataset containing basketball team names which are currently recorded in an inconsistent, mixed-case format. This scenario mirrors real-world data input challenges where data entry errors or varied sourcing lead to non-standard capitalization. Before applying the conversion functions, we must first establish the baseline dataset using the SAS DATA step and DATALINES statement.

The following code block creates a temporary dataset named `original_data`. This dataset includes a character variable called `team`, formatted to hold up to 20 characters. The input data intentionally exhibits variance in casing—ranging from entirely lowercase to mixed casing—setting the stage for our standardization efforts using UPCASE, LOWCASE, and PROPCASE.

```
/*create dataset*/
data original_data;
input team $1-20;
datalines;
Washington wizards
Houston rockets
Boston celtics
San antonio spurs
Orlando magic
Miami heat
;
run;

/*view dataset*/
proc print data=original_data;
```

Executing this code generates the initial dataset, which confirms the existence of non-standard capitalization across all team names. Observing the output from `PROC PRINT` helps visualize the problem we are aiming to solve, reinforcing the need for controlled case conversion operations to ensure data integrity and uniformity.

Obs	team
1	Washington wizards
2	Houston rockets
3	Boston celtics
4	San antonio spurs
5	Orlando magic
6	Miami heat

Example 1: Converting Strings to Uppercase for Consistency

The first practical step in data standardization often involves converting all textual identifiers to uppercase. This is generally preferred for consistency in reporting headers or when exporting data to systems that are case-insensitive or require fixed-format input. In this example, we apply the UPCASE function to the existing `team` variable within a new DATA step, effectively overwriting the original variable with its uppercase equivalent in the new dataset, `new_data`.

The structure utilizes the SET statement to read observations from the existing `original_data` and then applies the transformation formula: `team = UPCASE(team);`. This simple assignment statement ensures that as each record is processed, the value of the `team` variable is immediately standardized to capital letters before being written to the `new_data` dataset. Subsequently, the `PROC PRINT` step allows us to verify the successful transformation.

```
/*create new dataset*/
data new_data;
set original_data;
team = UPCASE(team);
run;

/*view new dataset*/
proc print data=new_data;
```

Upon reviewing the output dataset, every team name, such as "Washington wizards" or "San antonio spurs," has been accurately converted to all uppercase characters, resulting in "WASHINGTON WIZARDS" and "SAN ANTONIO SPURS." This outcome confirms the successful application of the **UPCASE** function for complete character standardization.

Obs	team
1	WASHINGTON WIZARDS
2	HOUSTON ROCKETS
3	BOSTON CELTICS
4	SAN ANTONIO SPURS
5	ORLANDO MAGIC
6	MIAMI HEAT

Example 2: Converting Strings to Lowercase for Text Analysis

In contrast to standardization for database identifiers, converting strings to lowercase is essential for many text processing and analytical tasks. By using the **LOWCASE** function, we strip away all capitalization, ensuring that all variations of a word are treated identically during frequency analysis or feature engineering. This example demonstrates how to apply the LOWCASE function to the `team` variable, maintaining a parallel structure to the previous example.

The code structure remains streamlined, leveraging the `DATA step` environment to process the records sequentially. The function call `team = LOWCASE(team);` ensures that the output dataset, `new_data`, contains the team names entirely in lowercase. This is highly useful for preparing textual data before employing more advanced SAS text analysis procedures.

```
/*create new dataset*/
data new_data;
set original_data;
team = LOWCASE(team);
run;

/*view new dataset*/
proc print data=new_data;
```

As observed in the resulting dataset, all team names are now consistently rendered in lowercase. For instance, "Houston rockets" remains "houston rockets," while "Boston celtics" is converted entirely to "boston celtics." This confirms the effectiveness of the **LOWCASE** function for uniform reduction of case variation.

Obs	team
1	washington wizards
2	houston rockets
3	boston celtics
4	san antonio spurs
5	orlando magic
6	miami heat

Example 3: Converting Strings to Proper Case for Reporting

When the goal is to produce output that is aesthetically pleasing and conforms to standard grammatical rules, such as official reports or mailing labels, the PROPCASE function is the optimal choice. Proper case ensures that only the initial letter of each distinct word is capitalized, a format traditionally used for proper nouns like names of people or organizations. It is important to remember that **Proper case** strictly defines the first letter of each word as capitalized, and all subsequent letters in that word as lowercase.

We apply the **PROPCASE** function to the `team` variable within the data step. This transformation is highly valuable for datasets where inconsistent capitalization must be resolved for presentation purposes without sacrificing human readability. Unlike UPCASE or LOWCASE, PROPCASE provides a balanced solution that respects conventional formatting.

```
/*create new dataset*/
data new_data;
set original_data;
team = PROPCASE(team);
run;

/*view new dataset*/
proc print data=new_data;
```

The final output dataset illustrates the precise effect of **PROPCASE**. Entries like "orlando magic" are transformed into "Orlando Magic," where both parts of the name are correctly capitalized. This showcases the function's ability to handle multi-word strings and achieve the desired proper case format quickly and reliably, solving common data quality issues related to presentation.

Obs	team
1	Washington Wizards
2	Houston Rockets
3	Boston Celtics
4	San Antonio Spurs
5	Orlando Magic
6	Miami Heat

Conclusion: Enhancing Data Quality through Case Conversion

The ability to reliably control the capitalization of string variables is an indispensable skill in the SAS programming toolkit. Whether your objective is rigorous data standardization using **UPCASE**, essential text normalization for analysis with **LOWCASE**, or polished, professional reporting via **PROPCASE**, these three functions provide the flexibility needed to handle diverse data quality requirements.

By integrating these functions into your daily DATA steps, you can ensure that your datasets maintain high levels of internal consistency. This consistency is not just a matter of aesthetics; it is a fundamental requirement for accurate data merging, reliable statistical comparisons, and the creation of reproducible analytical workflows. Proper case management is a small but powerful step towards maximizing the utility and trustworthiness of your enterprise data.

For those looking to deepen their expertise in character variable manipulation and other crucial data handling tasks within the SAS environment, further exploration of the extensive SAS documentation is highly recommended. Mastering these foundational functions allows programmers to move seamlessly onto more complex operations.