Learn How to Convert Strings to Uppercase, Lowercase, and Proper Case in SAS

Authored by

Mohammed looti

October 31, 2025

Introduction to String Case Conversion in SAS

The ability to manipulate the case of textual data, often referred to as strings, is fundamental to effective data cleaning and standardization. When working with large-scale datasets in SAS, inconsistencies in capitalization--such as names being entered in all caps, all lowercase, or mixed case--can severely complicate matching, merging, and reporting processes. Standardizing the case ensures data integrity and reliability across various analyses.

Fortunately, SAS provides three highly efficient and straightforward built-in functions specifically designed for this purpose: **UPCASE**, **LOWCASE**, and **PROPCASE**. These functions allow users to quickly transform entire variables within a dataset, making complex case standardization a simple one-line operation within the DATA step. Understanding how and when to apply each of these functions is crucial for maintaining a clean and analysis-ready dataset.

We will explore each function in detail, demonstrating their syntax and showing practical examples using a sample dataset containing various team names entered in non-standardized formats. This comprehensive guide will ensure you can efficiently manage case sensitivity in all your SAS programming tasks.

Core String Manipulation Functions

The three primary functions for case conversion in SAS operate on character variables and return a modified string based on the desired case transformation. Each method follows a simple syntax structure where a new variable is created or an existing one is overwritten by applying the function to the original string variable.

The following outlines the purpose and syntax of these essential SAS functions:

Method 1: Convert String to Uppercase (UPCASE)

The UPCASE function forces all characters within the input string to their uppercase equivalents. This is particularly useful when creating standardized identifiers or headings where consistency requires all capital letters.

new_string = UPCASE(old_string);

Method 2: Convert String to Lowercase (LOWCASE)

Conversely, the LOWCASE function transforms all characters in the input string to their lowercase forms. This is often applied during data preparation before using case-insensitive search or comparison operations, or simply for achieving a uniform appearance in text fields.

new_string = LOWCASE(old_string);

Method 3: Convert String to Proper Case (PROPCASE)

The PROPCASE function, also known as Title Case, capitalizes the first letter of every word within the string while converting all other letters to lowercase. This is the ideal function for standardizing names, titles, and addresses, ensuring professional and readable output. It effectively handles multi-word entries, such as "washington wizards," transforming them correctly into "Washington Wizards."

new_string = PROPCASE(old_string);

Setting Up the Sample Dataset

To effectively demonstrate the application of these functions, we will utilize a small sample dataset named `original_data`. This dataset simulates real-world data entry issues where team names have been input inconsistently, featuring mixed capitalization and all lowercase entries.

The following code employs the DATA step and `datalines` statement to create the initial dataset and then uses PROC PRINT to display the contents, allowing us to see the original, uncleaned state of the data.

```
/*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;
```

As observed in the output below, the 'team' variable contains mixed-case entries. For instance, "Washington wizards" has a capitalized first word but a lowercase second word, and "Boston

celtics" is entirely lowercase. This lack of standardization is precisely what the SAS case functions are designed to correct.

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

Example 1: Utilizing the UPCASE function

The most extreme form of case standardization involves converting all characters to uppercase. This format is frequently requested for specific reporting requirements, database key fields, or when preparing data for systems that are case-insensitive or require strict formatting. Using the UPCASE function allows us to quickly achieve this standard across the entire variable.

In the following code block, we initiate a new DATA step, setting the source dataset to `original_data`. We then apply the UPCASE function directly to the `team` variable, overwriting the original mixed-case values with their uppercase equivalents. This creates a new dataset, `new_data`, ready for review.

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

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

The resulting table clearly illustrates that every character in the `team` variable has been converted to uppercase, successfully standardizing the variable. For example, "Washington wizards" is now displayed as "WASHINGTON WIZARDS." This transformation ensures complete uniformity in the case formatting of the data.

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

Example 2: Implementing the LOWCASE function

Standardizing strings to lowercase is equally important, particularly in natural language processing (NLP) or when performing lookups against data sources that are strictly lowercase. Using the LOWCASE function ensures that all input, regardless of how it was originally entered, is converted to its lower-case representation.

We repeat the structure of the previous example, but substitute the LOWCASE function within the DATA step. This process demonstrates the flexibility of SAS string functions, which require only a minimal change in syntax to achieve a fundamentally different result. The use of the `set original_data` command ensures that we are always starting from the base, uncleaned dataset for each transformation.

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

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

Upon viewing the output generated by PROC PRINT, we can confirm that all team names, including those that previously contained initial capitals (like "Washington"), have been successfully converted entirely to lowercase. This provides a clean, standardized format ideal for certain data comparison tasks.

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

Example 3: Mastering the PROPCASE function

For data intended for human readability, such as reports, mailing lists, or general documentation, **Proper Case** (or Title Case) is usually the preferred format. The PROPCASE function automatically handles the complexities of multi-word strings, ensuring that the first letter of every significant word is capitalized while the rest remain lowercase. This provides the polished, professional look required for final reports.

Note: Proper case mandates that the initial letter of each word within the string be capitalized.

The application of PROPCASE function is identical in structure to the previous examples, requiring only the function name substitution. This demonstrates the high degree of consistency across the SAS function library, streamlining the learning curve for data analysts.

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

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

The output confirms the successful conversion to proper case. For instance, "Houston rockets" is now "Houston Rockets," and "San antonio spurs" is correctly formatted as "San Antonio Spurs." The PROPCASE function handles the capitalization of multiple words, which is often the most complex requirement for textual data standardization.

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

Additional Resources for SAS String Handling

Case conversion is just one component of robust string handling in SAS. Analysts frequently need to combine, split, search, and extract portions of strings to prepare data for modeling or reporting. Mastering the functions discussed here provides a strong foundation for exploring more advanced techniques.

For those looking to expand their capabilities beyond simple case transformation, the following areas are recommended for further study within the SAS documentation:

SUBSTR and INDEX functions: Essential for locating specific character sequences and extracting substrings based on position.
CAT functions (CATX, CATS, CATQ): Used for concatenating multiple character variables, often with specified delimiters, providing a powerful alternative to the simple double-bar operator (||).
TRIM and LEFT/RIGHT functions: Crucial for cleaning data by removing leading, trailing, or internal spaces that can interfere with matching operations.

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