How to Easily Define SAS Macro Variables with the %LET Statement

Authored by

stats writer

November 19, 2025

The %LET statement in SAS is a fundamental tool that empowers users to assign specific values to macro variables. This mechanism represents the most ubiquitous method for setting macro variables, capable of accommodating both character (string) and numeric data types. The strategic application of this statement is extensive; it facilitates the establishment of crucial default parameters, enables the efficient transfer of values across different segments of a program, and ensures the persistence of computed results for subsequent analytic tasks.

The syntax governing the %LET statement is straightforward: **%LET name = value;**. Here, 'name' designates the identifier of the new macro variable, while 'value' represents the precise data element being assigned. Mastering the appropriate use of %LET is crucial for writing dynamic and highly reusable SAS programs. Consult the official SAS documentation for a comprehensive collection of advanced %LET examples and use cases.

Understanding the Functionality of SAS Macro Variables

The core purpose of the **%LET** statement within the SAS Macro facility is to dynamically create and populate macro variables. These variables are distinct from standard SAS dataset variables; they exist within the macro environment and serve as symbolic substitutes that the SAS system resolves (replaces) before code execution. This capability is paramount for developing code that is adaptable, reusable, and less prone to manual configuration errors. When a program needs to reference a value that changes frequently--such as a cutoff date, a database name, or a processing threshold--using a macro variable ensures that updating the parameter requires only a single modification at the top of the script.

By employing the **%LET** statement, analysts gain significant flexibility in their programming workflow. Instead of hard-coding values directly into procedural steps or data manipulation logic, they can define critical parameters externally. This practice aligns with modern programming principles, promoting modularity and maintainability. When the SAS compiler encounters a reference to a macro variable (preceded by an ampersand, **&**), it substitutes the variable name with its assigned value before the DATA or PROC steps are executed, allowing the same code structure to operate seamlessly with different inputs.

Syntax and Practical Application of %LET

The basic structure of the %LET statement demands careful attention to naming conventions and data type handling. Variable names must adhere to standard SAS naming rules (starting with a letter or underscore, containing letters, numbers, or underscores, and being 32 characters or less). Unlike traditional assignment statements, the %LET statement does not typically require quotes around character values unless those quotes are intended to be part of the variable's value itself. However, when assigning string values for use in titles or complex code generation, careful

consideration of quotation marks is necessary, as demonstrated in later examples.

The crucial difference between assigning character and numeric values with %LET lies in how they are treated during macro resolution. For instance, when assigning a simple numeric value like 20 (as shown in Example 1), the assignment is straightforward. If the value contains special characters or spaces, it is generally treated as a character string. It is paramount that the assigned value resolves correctly into valid SAS code when referenced later in a program step, ensuring the subsequent DATA step or PROC step executes without error.

Example 1: Storing and Utilizing a Numeric Cutoff Value

A very common application of the **%LET** statement is defining a numeric threshold that dictates conditional processing within a SAS program. This approach allows analysts to easily adjust metrics--such as minimum performance scores, date ranges, or quantity limits--without altering the main processing logic. We will illustrate this by establishing a hypothetical performance standard for basketball players, where the macro variable will determine who qualifies as a "good player" based on their score.

Defining the Initial SAS Dataset

To demonstrate the utility of the macro variable, let us first define a sample dataset in SAS. This dataset, named **my_data**, contains essential information about various basketball players, including their respective teams and the points they scored in a recent match. The data is straightforward, designed to allow for easy testing of a numeric cutoff criteria.

```
/*create dataset*/
data my_data;
input team $ points;
datalines;
A 22
A 14
A 23
B 30
B 18
B 20
C 13
C 12
C 26
;
run;
```

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

Obs	team	points
1	A	22
2	A	14
3	A	23
4	B	30
5	B	18
6	B	20
7	C	13
8	C	12
9	C	26

Implementing %LET for Conditional Logic

We now proceed to utilize the %LET statement to define a performance benchmark. We create a macro variable named **points_cutoff** and assign it the numeric value of **20**. This variable acts as a configurable parameter for our analysis. Subsequently, we reference this macro variable within a new DATA step by preceding its name with the ampersand (**&**) symbol, which signals to the SAS system that macro resolution is required.

In the subsequent code, a new dataset (**new_data**) is generated, incorporating a flag variable called **good_player**. This variable uses the value stored in **&points_cutoff** to evaluate a logical condition: whether a player's score exceeds 20. If the condition is true, SAS assigns a value of 1; otherwise, it assigns 0. This demonstrates how a single, easily changeable macro variable can control complex conditional logic across the entire program structure.

```
/*assign value of 20 to macro variable*/
%let points_cutoff = 20;

/*use macro variable to create new column called good_player*/
data new_data;
set my_data;
good_player = points > &points_cutoff;
run;

/*view new dataset*/
```

proc print data=new_data;

Obs	team	points	good_player
1	A	22	1
2	A	14	0
3	A	23	1
4	B	30	1
5	B	18	0
6	B	20	0
7	C	13	0
8	C	12	0
9	C	26	1

Upon reviewing the output generated by the PROC PRINT statement, it is evident that the newly created column, **good_player**, accurately reflects the conditional logic based on the macro variable. Specifically, a value of **1** is assigned if the value in the **points** column strictly exceeds **20**. Conversely, if the points scored are less than or equal to **20**, the **good_player** flag is set to **0**. This result confirms the successful substitution and utilization of the numeric macro variable in data transformation.

Example 2: Using %LET for Dynamic String Assignment (Titles)

Beyond numeric values, the **%LET** statement is equally powerful for storing and utilizing character or string values. This is frequently employed to manage dynamic text elements, such as report titles, footnotes, or path specifications, providing a centralized point of control for text modification. When the analysis needs to be rerun with slightly different descriptive text, only the %LET statement needs adjustment, eliminating the need to search through many lines of procedural code.

In this second example, we will reuse the basketball dataset established previously, but focus on defining a customized title for the output report. This illustrates how macro variables simplify the aesthetic and descriptive aspects of statistical reporting within SAS.

Assigning a Dynamic Title String

The first step involves defining the character string we wish to use as the title. We use the **%LET** statement to create a macro variable called **table_title**. It is important to note that when assigning

text for titles that should appear in quotes within the resulting SAS code, the assignment itself must handle those quotes appropriately. In this case, we assign the value **"Basketball Data"**, ensuring the quotes are included so that when the macro variable is resolved, it generates valid syntax for the **TITLE** statement.

Executing the Titled Report

After the macro variable is defined, we utilize the PROC PRINT statement to display our data. Crucially, instead of writing the title text directly, we insert the macro reference **&table_title;** into the **TITLE** command. The SAS Macro Processor substitutes **&table_title;** with the assigned string **"Basketball Data"** before the procedure executes, resulting in a professional, dynamically generated title for the output.

```
/*create dataset*/
data my_data;
input team $ points;
datalines;
A 22
A 14
A 23
B 30
B 18
B 20
C 13
C 12
C 26
;
run;

/*assign string to macro variable*/
%let table_title = "Basketball Data";

/*print dataset with title*/
proc print data=my_data;
title &table_title;
run;
```

Basketball Data

Obs	team	points
1	A	22
2	A	14
3	A	23
4	B	30
5	B	18
6	B	20
7	C	13
8	C	12
9	C	26

As visible in the output, the resulting dataset report now proudly displays the custom title that was defined and managed through the **%LET** macro variable assignment. This confirms that **%LET** is an essential tool for managing both numeric thresholds and descriptive text strings efficiently within the SAS environment, drastically improving the flexibility of reporting mechanisms.

Key Best Practices for %LET Usage

To maximize the effectiveness and maintainability of SAS code utilizing the macro facility, several best practices related to the **%LET** statement should be followed. First, always define macro variables clearly at the beginning of the program or module where they are used. This enhances readability, allowing future users (or yourself) to immediately identify critical parameters. Secondly, employ clear and descriptive names for macro variables, avoiding vague or generic identifiers. For example, use **&REPORT_DATE** instead of just **&D**.

Furthermore, be mindful of the scope of your macro variables. Variables defined using **%LET** outside of any specific macro program are global and persist throughout the entire SAS session, making them available to all subsequent code blocks. If a variable is only needed temporarily within a macro definition, consider using the **%LOCAL** statement to prevent accidental overwriting of global variables. Finally, pay close attention to quoting rules, especially when assigning strings that contain semicolons, unmatched parentheses, or reserved SAS keywords, as these may require special handling using quoting functions like **%STR** or **%QSTR**, although simple strings often work without them.

Conclusion and Further Learning Resources

The **%LET** statement is undeniably one of the most critical components of dynamic programming in SAS. Its ability to centralize parameter management, support both numeric and character data, and seamlessly integrate with procedural steps makes it essential for developing robust and scalable analytic solutions. By leveraging macro variables for thresholds, conditional logic, and report customization, SAS users can significantly improve the efficiency and reusability of their codebases.

Understanding the subtle distinction between the macro environment and the data step environment, and recognizing how the ampersand triggers macro resolution, are key steps toward mastering the SAS Macro Language. We encourage users to practice these examples and explore additional functions that interact with the %LET statement, such as %PUT (for debugging) and %SYSEVALF (for complex numeric evaluations).

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