Learn How to Reorder Variables in SAS Datasets Using the RETAIN Statement

Authored by

Mohammed looti

October 31, 2025

In the world of statistical programming and data manipulation, the order in which variables appear within a dataset is often crucial for both clarity and subsequent processing. While the default behavior of the SAS (Statistical Analysis System) DATA step is to maintain the order in which variables are read or created, analysts frequently need to reorganize columns--perhaps to place identifier variables first, or to group related metrics for easier reporting.

Fortunately, SAS provides an elegant and efficient solution for this task: the **RETAIN** statement. Although its primary function is to hold a variable's value from one observation to the next, when used strategically within the DATA step structure, the **RETAIN** statement acts as a powerful tool for defining the output order of variables in a new dataset.

This tutorial will explore how the **RETAIN** statement can be leveraged specifically for variable reordering, covering the three most common structural methods employed by seasoned SAS programmers. We will provide detailed explanations and practical code examples demonstrating full reordering, as well as methods for simply promoting key variables to the front of the list while preserving the relative order of the remaining columns.

Understanding Variable Order in SAS

When SAS processes data, the order of variables often dictates the flow of reports and the ease of data interpretation. For example, in a financial dataset, having the 'CustomerID' and 'TransactionDate' variables appear immediately before metrics like 'Amount' or 'Status' significantly improves readability. Conversely, if variables are scattered arbitrarily, the dataset becomes cumbersome to use, especially when dealing with hundreds of columns.

By default, when you use a **SET** statement to copy an existing dataset, the new dataset inherits the exact variable order of the source dataset. To override this default arrangement, we must intervene in the compilation phase of the SAS DATA step. This is where the **RETAIN** statement proves invaluable, as it is processed before the data records are actually read.

The key principle lies in placing the **RETAIN** statement before the **SET** statement. When SAS encounters **RETAIN** listing specific variables, it defines and sets aside space for these variables first. Crucially, if the variables listed in **RETAIN** already exist in the input dataset specified by the **SET** statement, SAS defines their position in the new output dataset according to the order listed in the **RETAIN** statement, regardless of their original position.

The Power of the RETAIN Statement for Reordering

The **RETAIN** statement serves two main purposes in SAS programming. Its most common function is persistence--telling SAS to retain the value of a variable from the previous observation for the current observation. However, when used solely to list variables that exist in the input dataset and

placed strategically before the **SET** statement, its function morphs entirely into a variable ordering mechanism.

The flexibility of this approach allows developers to execute precise control over the output structure without resorting to more complex procedures like the **PROC DATASETS** or **ATTRIB** statements, which, while powerful, often involve more intricate syntax or require working outside the main DATA step flow. Utilizing **RETAIN** is often considered the cleanest and most direct method for restructuring variables dynamically within a data flow.

Here are the three fundamental methods for variable reordering using the **RETAIN** statement, which we will demonstrate fully in the examples below:

Method 1: Reorder All Variables (Full Specification)

This method requires explicitly listing every variable in the exact desired sequence. Any variable not listed will be omitted from the output dataset, emphasizing the need for precision when using this technique for complete restructuring.

```
data new_data;
retain var4 var5 var1 var3 var2;
set original_data;
run;
```

Method 2: Move One Variable to Front (Partial Specification)

By listing only a subset of variables, we instruct SAS to place these specified variables at the beginning of the new dataset. All variables *not* listed in the **RETAIN** statement will follow in their original order from the source dataset, making this a quick way to promote key variables.

```
data new_data;
retain var4;
set original_data;
run;
```

Method 3: Move Several Variables to Front (Group Promotion)

Similar to Method 2, this technique allows for the promotion of multiple variables to the starting positions. The order of these promoted variables will strictly follow the sequence defined in the **RETAIN** statement, while the remaining variables maintain their initial relative order.

```
data new_data;
```

```
retain var4 var5;
set original_data;
run;
```

Practical Demonstration: Setting Up the Sample Dataset

To illustrate these reordering methods, we will utilize a sample dataset containing basic basketball statistics. This dataset, named `original_data`, includes five variables: `team` (character), `points`, `rebounds`, `assists`, and `steals` (numeric). Notice the initial order of these variables, as it is this sequence that we aim to manipulate in the subsequent examples.

The setup code below creates and displays the initial structure of our sample data. This foundation is essential for observing the changes introduced by the **RETAIN** statement in each method.

```
/*create dataset*/
data original_data;
input team $ points rebounds assists steals;
datalines;
A 18 10 4 5
B 24 11 6 7
C 26 14 6 8
D 34 22 5 3
E 38 3 7 7
F 45 12 4 4
G 23 7 9 1
;
run;

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

Obs	team	points	rebounds	assists	steals
1	A	18	10	4	5
2	B	24	11	6	7
3	C	26	14	6	8
4	D	34	22	5	3
5	E	38	3	7	7
6	F	45	12	4	4
7	G	23	7	9	1

As displayed by the output of the PROC PRINT step, the default variable order is: `team`, `points`, `rebounds`, `assists`, and `steals`. We will now proceed to reorganize these columns using the three distinct applications of the **RETAIN** statement.

Example 1: Specifying a Complete Custom Order

When the requirement is a total overhaul of the dataset structure, Method 1--listing every variable in the **RETAIN** statement--is necessary. This ensures that the new dataset, `new_data`, will contain all original variables, but positioned precisely according to the specified sequence.

For this example, suppose we want to place the identifier variable (`team`) first, followed by defensive statistics (`rebounds`, `assists`, `steals`), and finally, the primary offensive statistic (`points`). The desired sequence is: `team`, `rebounds`, `assists`, `steals`, then `points`.

The following code block demonstrates how to achieve this complete reordering. Notice that the variables are listed in the **RETAIN** statement in the exact desired output sequence. Since all variables from `original_data` are listed in **RETAIN**, the new dataset structure is entirely defined by this statement.

```
/*create new dataset with variables reordered*/
data new_data;
retain team rebounds assists steals points;
set original_data;
run;

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

Obs	team	rebounds	assists	steals	points
1	A	10	4	5	18
2	B	11	6	7	24
3	C	14	6	8	26
4	D	22	5	3	34
5	E	3	7	7	38
6	F	12	4	4	45
7	G	7	9	1	23

The resulting dataset, `new_data`, successfully reflects the specified sequence: `team`, `rebounds`, `assists`, `steals`, and `points`. This confirms that the **RETAIN** statement, when used for full specification, provides absolute control over the final variable arrangement.

Example 2: Promoting Single Key Variables

Often, a programmer only needs to promote one or two critical variables to the very beginning of the dataset without disrupting the sequence of the remaining columns. Methods 2 and 3 address this scenario by utilizing the implicit ordering behavior of SAS.

When the **RETAIN** statement lists only a subset of the total variables, SAS first places the retained variables in the specified order. Then, it automatically appends all remaining variables from the source dataset in their original relative order. This is highly efficient when dealing with datasets that have many variables, where listing every single column would be impractical or prone to error.

In this example, let's assume we only want the `assists` variable to be the very first column, leaving all other variables (`team`, `points`, `rebounds`, `steals`) to follow exactly as they were ordered in `original_data`.

```
/*create new dataset with variables reordered*/
data new_data;
retain assists;
set original_data;
run;

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

Obs	assists	team	points	rebounds	steals
1	4	A	18	10	5
2	6	B	24	11	7
3	6	C	26	14	8
4	5	D	34	22	3
5	7	E	38	3	7
6	4	F	45	12	4
7	9	G	23	7	1

Upon reviewing the output, we clearly observe that `assists` is now the first variable. The remaining variables--`team`, `points`, `rebounds`, and `steals`--follow, maintaining the same sequential relationship they had in the input dataset, but shifted to accommodate the promoted variable. This partial use of the **RETAIN** statement provides excellent streamlining for data preparation tasks.

Example 3: Moving Multiple Variables to the Front

Building on the previous method, we can promote a group of variables to the front of the dataset. The order of the promoted group is determined strictly by their listing in the **RETAIN** statement, while the rest of the variables maintain their original relative sequence. This is Method 3: promoting several key variables simultaneously.

Let's demonstrate how to move both the `assists` and `rebounds` variables to the front. We want `assists` to be first and `rebounds` to be second. All other variables should follow in their original order. Note that the order within the **RETAIN** statement is critical; listing `assists` before `rebounds` ensures that `assists` takes the primary position.

```
/*create new dataset with variables reordered*/
data new_data;
retain assists rebounds;
set original_data;
run;

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

Obs	assists	rebounds	team	points	steals
1	4	10	A	18	5
2	6	11	B	24	7
3	6	14	C	26	8
4	5	22	D	34	3
5	7	3	E	38	7
6	4	12	F	45	4
7	9	7	G	23	1

The resulting output confirms that `assists` is now in the first position, followed by `rebounds` in the second position. The remainder of the variables--`team`, `points`, and `steals`--follow, preserving their original order relative to one another. This demonstrates the efficiency of using **RETAIN** for focused variable promotion without needing to manually account for every single column in the dataset structure. Understanding the difference between full specification (Example 1) and partial specification (Examples 2 and 3) is key to writing clean and maintainable SAS code.

Additional Resources and Further Study

Mastering the **RETAIN** statement for variable reordering is a fundamental skill in SAS programming, significantly enhancing data management efficiency. While this technique is robust, SAS offers several other powerful methods and procedures for structuring and manipulating data.

For those looking to deepen their expertise, exploring alternative methods such as using the **LENGTH** or **ATTRIB** statements in conjunction with the DATA step, or employing the **PROC DATASETS** procedure for permanent changes to a library's metadata, is highly recommended. These tutorials provide further insight into common advanced tasks in SAS:

How to manage variable attributes using the **ATTRIB** statement.
Techniques for conditionally dropping or keeping variables in a dataset.
Advanced uses of the **RETAIN** statement for cumulative calculations.