How to Easily Reorder Variables in SAS Datasets

Authored by

stats writer

December 1, 2025

Managing and manipulating data structures efficiently is a cornerstone of effective statistical programming. In the realm of the SAS System, reordering the sequence of variables within a dataset is a frequently required technique. This practice is not merely cosmetic; it significantly enhances data readability, streamlines report generation, and simplifies subsequent analyses by grouping related fields logically.

While SAS processes variables alphabetically or in the order they were first encountered during compilation by default, users often need a custom arrangement. Historically, methods such as using the DATA step combined with statements like RENAME=, DROP=, or KEEP= were employed, sometimes requiring tedious manual specification. However, the most robust and elegant solution in modern SAS programming involves leveraging the power of the RETAIN statement.

This comprehensive guide details how to master variable reordering using the RETAIN statement within a DATA step, offering practical examples ranging from fully customized reordering to simply moving one or more key variables to the forefront of the structure. Understanding these techniques is essential for any professional working with large or complex SAS datasets, ensuring maximum efficiency and clarity in data management workflows.

The Power of the RETAIN Statement for Reordering

The RETAIN statement is primarily known for preserving the values of variables across iterations of the DATA step, preventing them from being reset to missing. However, it possesses a powerful side effect: when the RETAIN statement is placed before the SET statement in a DATA step, SAS recognizes the listed variables and determines their position in the new output dataset based on the order in which they appear in the RETAIN list.

This mechanism offers a straightforward, declarative way to control the output order without needing complex procedural code. When you specify a list of existing variables in the RETAIN statement, SAS first creates those variables in the specified sequence. Subsequently, when the SET statement reads the input dataset, any variables not explicitly mentioned in the RETAIN statement are appended to the end of the new dataset in their original sequence from the source. This duality allows for both complete customization and partial reordering strategies.

Mastering this technique is crucial because it significantly reduces the risk of error compared to using KEEP= statements, which must list every single variable, or repeated RENAME= statements, which can clutter the code. By using RETAIN, the variable list only needs to include the variables you wish to reposition, making maintenance simpler and the code far more readable. The following sections demonstrate the three primary patterns for leveraging this essential SAS function.

Here are the three most common ways to use this function:

Method 1: Reorder All Variables

This method requires listing every variable in the desired output sequence.

Method 2: Move One Variable to Front

List the single variable first; all others will follow automatically in their original relative order.

Method 3: Move Several Variables to Front

List the subset of variables you want at the start, followed by the SET statement.

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

When executing this code, SAS checks the structure of the input dataset, `original_data`. It identifies `var4`, `var5`, `var1`, `var3`, and `var2`, and reserves their positions exactly as listed in the `RETAIN` statement. Any other variables present in `original_data` but omitted from the `RETAIN` list would automatically be placed after `var2`, maintaining their original relative order among themselves. This technique provides the flexibility needed for various data presentation requirements.

Setting Up the Example Dataset in SAS

To provide tangible examples of variable reordering, we will utilize a sample dataset representing statistical records for a sports league. This dataset, named `original_data`, contains five variables covering team identification and four key performance metrics. Understanding the initial structure is crucial before attempting any reordering transformations, as it establishes the baseline against which the changes are measured.

The initial structure features the variables in the sequence they were defined: `team`, `points`, `rebounds`, `assists`, and `steals`. Our goal in the subsequent examples will be to reposition these variables to prioritize different metrics--for instance, placing defensive statistics before offensive statistics--to better suit a specific analytical focus or reporting requirement.

We use the following DATA step code, including the DATALINES statement, to create and populate this source dataset, followed by a simple `PROC PRINT` to view the initial arrangement before applying any reordering logic.

```
/*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;
```

As confirmed by the output, the variables currently follow the input order, which might not be ideal if the reporting focus shifts away from `points` to other metrics like `assists` or `rebounds`. The following image illustrates the default order.

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

Example 1: Comprehensive Reordering of All Variables

When the requirement is to establish a completely new, custom order for every variable in the dataset, the RETAIN statement must explicitly list every single variable in the precise sequence desired. This approach ensures that no variables are left to default placement and provides the most stringent control over the final data structure. It is particularly useful when preparing data for fixed-format reports or external systems that mandate a specific column arrangement.

For this example, imagine we are shifting focus from offensive metrics to a structure that emphasizes identifiers and defensive/support statistics first. We aim for the following sequence: `team`, `rebounds`, `assists`, `steals`, and finally `points`. By placing the `RETAIN` statement at the beginning of the DATA step, we preempt the standard variable creation process and enforce our custom order.

The following code shows how to reorder the variables in the desired sequence: `team`, `rebounds`, `assists`, `steals`, and then `points`. Notice how the RETAIN statement mirrors this target order precisely.

```
/*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;
```

After executing this code, the resulting output dataset, `new_data`, displays the variables in the exact order specified by the `RETAIN` function. This confirms the efficacy of the method for total structural control. If any variables were added to `original_data` later, they would have been ignored in the output dataset unless explicitly listed in the `RETAIN` statement, highlighting the importance of comprehensive listing when full control is needed.

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

Notice that the variables are reordered in the exact order that we specified in the **RETAIN** function, with `rebounds` now occupying the second column position, followed by `assists` and `steals`.

Example 2: Strategically Moving a Single Variable to the Front

Often, the requirement is not to reorder the entire structure but simply to bring one or two critical variables forward for immediate visibility, leaving the relative order of all other fields intact. This partial reordering technique is far more efficient than listing every variable manually, especially in datasets containing dozens or hundreds of columns.

The key to this method lies in the dual function of the RETAIN statement: it defines the order of listed variables first, and then the subsequent `SET` statement reads the input data, appending all unlisted variables in their original sequence. By listing only the desired front-runner variable, we ensure it takes the first position while the rest of the structure remains minimally disturbed.

In this scenario, let us say we want the `assists` variable to be the very first field after the `team` identifier. Since the `team` variable is typically the first field, we will list only the `assists` variable in the RETAIN statement. SAS automatically maintains the identity variable (`team`) in the first position if it is not explicitly moved, but for clarity, we should generally include the variables we wish to control.

However, the most economical way to move a variable to the front while preserving all others is to simply list that variable. Since the original data step created `team` first, and then `points`, `rebounds`, `assists`, and `steals`, listing only `assists` will push it to the front, and the remaining variables will follow in their original relative order (`team`, `points`, `rebounds`, `steals`).

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

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

The output below confirms the result: `assists` has been successfully moved to the second column position (after `team`), while `points`, `rebounds`, and `steals` follow precisely in their original relative sequence. This technique is highly valuable for rapid data exploration and validation within SAS environments.

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

We can see that the **assists** variable is now in the first position (following the default position of the `team` variable, which is usually retained as the first variable if not specified otherwise, or explicitly defined as the first position if `team` was character and needed length defined) while all of the other variables remained in the same order relative to each other.

Example 3: Positioning Multiple Variables at the Start

Extending the concept from Example 2, we can easily move a small group of related variables to the beginning of the dataset structure without disrupting the order of the remaining fields. This is accomplished by simply listing all desired front-runners sequentially in the RETAIN statement, followed by the `SET` statement which pulls in the rest of the variables.

For instance, if a report needs to immediately show the support statistics (`assists` and `rebounds`) right after the team identifier, we would list these two variables in the desired order within the `RETAIN` statement. All variables not explicitly mentioned in the statement will then populate the remaining columns, preserving their original relative positions from the source dataset.

This method is powerful because it combines granular control over the most important fields with the automatic handling of less critical or static fields. It represents a balanced approach between the manual specificity of Example 1 and the minimal effort of Example 2.

The following code shows how to move the **assists** and **rebounds** variables to the front while leaving all other variables in the same order:

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

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

When this DATA step executes, `assists` is placed first, followed by `rebounds`. The remaining variables (`points` and `steals`, along with `team`) are appended at the end of the list in their pre-existing sequence relative to each other. Note that `team`, being the key identifier, is often implicitly first or its position determined by the `RETAIN` list if we include it, but here, it defaults to the first position.

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

We can see that the **assists** and **rebounds** variables are now in the second and third positions (following the `team` variable), while all of the other variables remained in the same order, illustrating perfect control over the initial columns.

Advanced Considerations and Alternative Methods

While the RETAIN statement is the preferred method for reordering variables within a dataset in SAS, it is important to understand alternative techniques and crucial technical considerations, particularly concerning new variables or variable attributes.

One major consideration when using `RETAIN` is handling new variables. If you introduce a new variable (one not present in the input dataset), the `RETAIN` statement will also define its position. However, if you are creating new variables that require specific formatting or length definitions, it is best practice to define those attributes using the `LENGTH` or `FORMAT` statements alongside `RETAIN`, and place all of these definitions prior to the `SET` statement to ensure proper definition and positional placement.

An older, less flexible but still valid alternative to the `RETAIN` statement is the use of the `KEEP=` or `DROP=` dataset options on the `SET` statement, coupled with a specific variable ordering in a prior `LENGTH` statement. The `LENGTH` statement, like `RETAIN`, assigns an order, but unlike `RETAIN`, it is

explicitly designed to define variable attributes and is not commonly used solely for reordering existing variables due to potential conflicts with incoming data attributes.

For advanced users managing libraries with many datasets, `PROC DATASETS` offers a non-data step approach to variable reordering. Using the `MODIFY` statement along with the `REORDER` option allows users to change the variable order metadata directly without rewriting the entire dataset. This method is highly efficient for very large files where minimizing data processing time is paramount, although it requires precise knowledge of the target order and is less intuitive than the `RETAIN` approach used within the DATA step.

The choice of method should align with the complexity of the task: `RETAIN` for flexibility and readability in a data flow, and `PROC DATASETS` for efficiency when dealing with huge, static datasets where only the structural metadata needs updating.

Summary of Variable Ordering in SAS

Effective variable reordering is a fundamental skill in SAS programming, optimizing data presentation and simplifying downstream analysis. The RETAIN statement provides the most streamlined and readable solution for this task within the DATA step, offering precise control over the positional placement of existing variables.

By placing the `RETAIN` statement before the `SET` statement, you dictate the structure of the new output dataset. We have explored three crucial patterns: listing all variables for total structural customization, listing one variable for targeted repositioning to the front, and listing a subset of variables to group them at the start while preserving the relative order of the remainder. Each approach serves a distinct analytical or reporting requirement.

Consistent application of the `RETAIN` technique ensures that your data preparation is both robust and easily auditable. For future reference and to expand your SAS proficiency, consider exploring other complementary data manipulation techniques.

Further SAS Tutorials

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

How to Merge Datasets in SAS
Understanding PROC SQL for Data Manipulation
Using the ARRAY Statement in SAS