How to Use PROC SORT and the KEEP Statement in SAS to Select Specific Data

Authored by

stats writer

November 20, 2025

As an experienced user of the SAS system, mastering efficient data manipulation techniques is crucial for effective statistical analysis. One powerful combination often utilized for simultaneously ordering and refining datasets is the pairing of the PROC SORT procedure with the KEEP statement. While `PROC SORT` is primarily designed to order observations based on specified criteria, integrating the `KEEP` statement allows the user to define precisely which columns, or variables, are carried forward into the resulting output data set.

The `KEEP` statement, when applied as a data set option within the `OUT=` parameter of `PROC SORT`, serves as a vital gatekeeper. It enables analysts to streamline their data by retaining only the variables pertinent to subsequent analyses, thereby reducing memory usage and processing time. This strategy ensures that complex procedures or large datasets do not retain unnecessary metadata or columns, leading to a much cleaner and more focused analytical environment. Understanding the syntax and proper application of this technique is fundamental for generating optimized and well-formed SAS output.

It is important to clarify a common point of confusion: the `KEEP` statement used in this context specifies which variables (columns) are retained, not which observations (rows) are selected based on values. Observation selection is typically managed using the `WHERE` statement or filtering steps prior to or within a `DATA` step. By defining the list of variables to `KEEP`, we ensure that the structure of the resulting sorted dataset is minimized to include only essential components, making the resulting dataset lighter and more manageable for immediate use or long-term storage within the SAS library ecosystem.

Understanding Data Subset Management in SAS

Effective data management often necessitates the creation of subsets from larger, comprehensive datasets. This process is critical for focusing analytical efforts, reducing computational load, and adhering to privacy or data usage protocols. In SAS, there are multiple methods for creating subsets, generally falling into two categories: subsetting observations (rows) and subsetting variables (columns). When analysts work with massive databases, the ability to selectively choose which columns are retained during a major operation like sorting becomes a significant factor in performance optimization.

While a standard `DATA` step allows for precise control over both row and column subsetting using `WHERE`, `KEEP`, or `DROP` statements, integrating these functions directly into procedural steps, where possible, often yields cleaner code and better execution efficiency. The specific combination of `PROC SORT` with the `KEEP statement` fulfills this need by enabling simultaneous reordering and column pruning. This method is particularly valued when the input data set is extremely wide, containing hundreds of variables, but only a handful are required for the immediate analytical task.

The Core Functionality of PROC SORT

The `PROC SORT` procedure is one of the most fundamental and heavily utilized procedures in the SAS analytical toolkit. Its primary purpose is to reorder the observations within a data set based on the values of one or more specified sorting variables. Proper sorting is essential for numerous subsequent operations, including merging datasets, grouping data for summary statistics (using `BY` groups in other procedures like `PROC MEANS` or `PROC PRINT`), and ensuring data integrity for time-series analysis.

When executing `PROC SORT`, the output dataset, specified by the `OUT=` option, typically inherits the entire structure of the input dataset, including all variables, unless otherwise instructed. By default, sorting is performed in ascending order, though this can be reversed using the `DESCENDING` keyword before the variable name in the `BY` statement. While sorting is the core function, the procedure's flexibility is significantly enhanced by its ability to accept data set options on the `OUT=` parameter, such as the `KEEP` option, transforming it into a combined reordering and data refinement tool.

Integrating the KEEP Statement for Variable Selection

The `KEEP statement`, when used as a data set option within the `PROC SORT` output, provides a concise and elegant way to specify the exact subset of variables that should be written to the new, sorted dataset. This is far more efficient than sorting the entire dataset and then following up with a separate `DATA` step solely to drop unwanted columns. The syntax allows the user to list variables explicitly, saving processing time and disk space immediately upon creation of the new dataset.

When applying the `KEEP` option, it is paramount that any variable used in the mandatory `BY` statement must also be included in the list of variables specified for `KEEP`. If a variable used to define the sort order is inadvertently excluded from the `KEEP` list, SAS will generate an error because the structure necessary for the sorting operation would be incomplete in the output. This requirement reinforces the logical relationship between the sorting criteria and the resulting dataset structure, ensuring that the necessary contextual variables for the sorted order are always present in the final output. This powerful utility allows analysts to focus on only the dataset features required for subsequent modeling or reporting tasks.

Deconstructing the Essential SAS Syntax

To leverage the combined power of sorting and subsetting, you must utilize the following generalized syntax structure. This structure ensures that `PROC SORT` executes its primary function (sorting) while simultaneously applying the variable filtering defined by the KEEP statement within the output options. This integration streamlines the code and enhances processing efficiency

compared to sequential steps.

The fundamental template requires specifying the input dataset, defining the output dataset with the embedded `KEEP` option, and finally, declaring the variable(s) by which the sorting will occur using the `BY` statement. Note that the `KEEP` parameter is contained within parentheses immediately following the `OUT=` dataset name, distinguishing it as a data set option.

```
proc sort data=my_data out=sorted_data (keep=var1 var2);
by var2;
run;
```

In this specific example, the procedure instructs SAS to sort the rows of the dataset `my_data` based on the values found in `var2`. Crucially, the resulting dataset, named `sorted_data`, will only contain two variables: `var1` and `var2`. Any other variables present in the original `my_data` dataset will be immediately excluded from the output, ensuring a concise and focused result set tailored for subsequent analysis.

Practical Demonstration Setup: Creating the Dataset

To illustrate the practical application of `PROC SORT` with the `KEEP statement`, we will establish a simple sample data set containing information about various professional basketball teams. This dataset includes three crucial variables: `team` (character variable denoting the team name), `points` (numeric variable for average points scored), and `assists` (numeric variable for average assists).

The initial step involves using a `DATA` step and the `DATALINES` statement to input the raw data directly. This ensures that we have a clean, unsorted starting point, `my_data`, which contains all three variables. Subsequently, we use `PROC PRINT` to verify the structure and content of our newly created dataset, confirming that all observations and variables are present before we begin the sorting and subsetting operations. This baseline verification is an important practice in any data manipulation workflow.

```
/*create dataset*/
data my_data;
input team $ points assists;
datalines;
Mavs 113 22
Pacers 95 19
Cavs 100 34
Lakers 114 20
Heat 123 39
```

Kings 100 22
Raptors 105 11
Hawks 95 25
Magic 103 26
Spurs 119 29
;
run;

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

Obs	team	points	assists
1	Mavs	113	22
2	Pacers	95	19
3	Cavs	100	34
4	Lakers	114	20
5	Heat	123	39
6	Kings	100	22
7	Raptors	105	11
8	Hawks	95	25
9	Magic	103	26
10	Spurs	119	29

Scenario 1: Standard Sorting (Retaining All Variables)

Before applying the subsetting capabilities, it is useful to establish a baseline by performing a standard sort operation. This demonstrates how `PROC SORT` behaves when no variable selection options are specified on the output dataset. We will sort the `my_data` dataset based on the `points` variable, creating a new dataset called `sorted_data`.

When the `OUT=` option is used without additional data set options, the resultant dataset retains all variables from the source dataset. This outcome confirms the default behavior of SAS, where sorting only changes the physical order of the observations but preserves the complete column structure. This step is crucial for comparison against the scenario where we introduce the selective `KEEP` statement.

```
/*sort rows in dataset based on values in points column*/
proc sort data=my_data out=sorted_data;
by points;
run;

/*view sorted dataset*/
proc print data=sorted_data;
```

Obs	team	points	assists
1	Pacers	95	19
2	Hawks	95	25
3	Cavs	100	34
4	Kings	100	22
5	Magic	103	26
6	Raptors	105	11
7	Mavs	113	22
8	Lakers	114	20
9	Spurs	119	29
10	Heat	123	39

Upon reviewing the output, it is clear that the rows are now logically arranged in ascending order based on the values in the `points` column. Furthermore, all three original variables--`team`, `points`, and `assists`--have been retained in the `sorted_data` dataset. This demonstrates the standard, comprehensive output of the sorting procedure when variable subsetting is not explicitly applied.

Scenario 2: Selective Variable Retention using KEEP

Now, we introduce the core technique: using the `KEEP statement` as a data set option during the `PROC SORT` operation. Suppose our subsequent analysis only requires the team name and their corresponding average points, but the `assists` variable is unnecessary. We can modify the `OUT=` statement to include the `KEEP` option, listing only `team` and `points`.

This implementation achieves the sort operation based on `points` while simultaneously performing variable subsetting. The list of variables provided to `KEEP` dictates the final structure of the output dataset, ensuring that only those specified columns are written to the disk. This results in a cleaner, more focused data set that is immediately ready for targeted analysis, significantly streamlining the workflow.

```
/*sort rows in dataset based on values in points column and only keep team and points*/
proc sort data=my_data out=sorted_data (keep=team points);
by points;
run;

/*view sorted dataset*/
proc print data=sorted_data;
```

Obs	team	points
1	Pacers	95
2	Hawks	95
3	Cavs	100
4	Kings	100
5	Magic	103
6	Raptors	105
7	Mavs	113
8	Lakers	114
9	Spurs	119
10	Heat	123

The resulting output confirms that the data has been sorted correctly based on `points` in ascending order. More importantly, only the `team` and `points` variables were carried over to the `sorted_data` dataset, demonstrating the successful application of the `KEEP` data set option. The `assists` variable was efficiently dropped during the sorting process, achieving the desired reduction in dataset size and complexity immediately.

Efficiency and Advantages of In-Procedure Selection

Using the `KEEP statement` directly within the `PROC SORT` output option offers significant advantages in terms of coding efficiency and performance, especially when dealing with large datasets typical in enterprise SAS environments. The primary benefit is the reduction in required processing steps; combining sorting and variable subsetting into a single procedure call eliminates the need for a separate subsequent `DATA` step solely dedicated to dropping columns.

From a performance perspective, SAS only writes the specified variables to the new output file on disk. If the original dataset is very wide (e.g., 500 variables) but only 10 are needed, the I/O operations are dramatically reduced, saving disk space and speeding up the overall execution time

of the job. This streamlined approach minimizes overhead and ensures that computational resources are focused only on the data elements that hold analytical value. Furthermore, concise code is inherently less prone to errors and is easier for other analysts to audit and maintain.

Advanced Considerations and Best Practices

While the `KEEP` data set option is highly effective for variable retention, analysts should also be aware of related options and best practices. The inverse operation, variable exclusion, can be accomplished using the `DROP` data set option, which functions identically to `KEEP` but specifies variables to exclude rather than include. For instance, `(DROP=var3 var4)` would keep all variables except `var3` and `var4`.

It is also possible to rename variables simultaneously while sorting and subsetting using the `RENAME` data set option within the same parentheses as `KEEP`. For example: `(KEEP=team points RENAME=(points=avg_points))`. When structuring complex data procedures, always prioritize using `KEEP` over `DROP` when the number of variables to retain is significantly smaller than the total number of variables in the dataset. This minimizes the risk of accidentally omitting a necessary column if the structure of the source dataset changes over time.

Conclusion and Further Resources

The strategic deployment of the `KEEP statement` within the `OUT=` option of PROC SORT represents an essential technique for efficient data preparation in SAS. By allowing analysts to simultaneously order observations and precisely define the resulting column structure, this method ensures that datasets are streamlined, optimized for size, and perfectly aligned with the needs of subsequent analytical procedures. This not only improves processing speed but also contributes significantly to the clarity and maintainability of SAS programs, which is vital in professional data environments.

Mastering this simple syntax distinction--using `KEEP` as a data set option--is key to moving beyond basic SAS programming toward advanced efficiency. We have demonstrated how this integration yields a clean, sorted, and subsetted output data set in a single, powerful step, confirming its status as a best practice for managing large and complex data structures.

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