Understanding PROC PRINT in SAS: A Comprehensive Tutorial with Examples

Authored by

Mohammed looti

November 14, 2025

The Power and Purpose of PROC PRINT

The PROC PRINT procedure stands as one of the most fundamental and frequently utilized tools within the SAS programming environment for visualizing the contents of a dataset. This procedure is absolutely essential not only for verifying data integrity and conducting quick quality checks but also for generating structured reports that confirm data manipulation steps have been executed correctly. While its basic application--displaying raw data--is exceedingly simple, PROC PRINT provides a powerful suite of optional statements for robust customization.

Mastery of these customization options allows users granular control over the final report output. Users can precisely dictate which observations (rows) and variables (columns) are displayed, how the resulting output is logically grouped based on categorical values, and how the overall report is formatted for readability. Understanding these optional statements is the key to transforming raw data dumps into professional, tailored reports suitable for documentation or stakeholder review.

This guide explores five essential methods for leveraging the full potential of PROC PRINT. By systematically applying these techniques, developers and analysts can establish a strong foundation for effective data presentation and reporting within the complex SAS environment. We will utilize practical code examples to demonstrate how to implement these refinements effectively.

Five Essential Methods for Customized Reporting

The following list outlines the core capabilities we will explore in detail. These methods move beyond the default functionality of **PROC PRINT**, enabling you to tailor reports precisely to your analytical requirements:

Method 1: Printing the Entire Dataset: Utilizing the most basic syntax to display all rows and columns, serving as the default foundation for all subsequent methods.
Method 2: Limiting Observations (N): Applying the `OBS=` option to restrict the output to a specified number of initial rows, which is crucial for quick data verification and debugging large datasets.
Method 3: Selecting Specific Variables: Controlling column visibility and their display order using the crucial VAR statement.
Method 4: Grouping Reports by Variable: Generating segmented reports based on unique categorical values using the BY statement, which requires prior sorting of the data.
Method 5: Adding Descriptive Context: Enhancing the report's professional appearance and readability by incorporating custom titles and footers using the global `TITLE` and `FOOTNOTE` statements.

Preparing the Sample Data for Demonstration

To ensure practical demonstration of these reporting techniques, we will first create and utilize a sample SAS dataset named `my_data`. This sample data simulates a small collection of basketball player statistics, featuring both categorical variables (like `team` and `position`) and numerical metrics (such as `points` and `assists`). This mixed data structure allows us to effectively test grouping and variable selection capabilities.

Before proceeding to the functional examples of **PROC PRINT**, you must ensure that this sample dataset is successfully loaded and available within your active SAS session. The following Data Step provides the necessary code to create and populate the `my_data` table, serving as the foundational input for all subsequent procedure calls.

```
/*create dataset*/
data my_data;
input team $ position $ points assists;
datalines;
A Guard 14 4
A Guard 22 6
A Guard 24 9
A Forward 13 8
A Forward 13 9
A Guard 10 5
B Guard 24 4
B Guard 22 6
B Forward 34 2
B Forward 15 5
B Forward 23 5
B Guard 10 4
;
run;
```

Once this code block has been executed, the `my_data` dataset will reside in your working library, ready to be processed by the various configurations of the **PROC PRINT** procedure demonstrated in the following methods.

Method 1 & 2: Controlling Data Volume (Full Print and OBS=)

The most fundamental use of **PROC PRINT** involves displaying every single observation (row) and every variable (column) present within the specified input dataset. This default behavior is

executed by simply providing the procedure name followed by the `DATA=` option, which points to the source table. This basic syntax is often the first step in confirming data structure and initial load success.

The code below illustrates the standard execution required to print the complete contents of the `my_data` dataset. Note that without any optional statements, **PROC PRINT** automatically generates an observation number column on the left side of the report, ensuring that every record is indexed.

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

Obs	team	position	points	assists
1	A	Guard	14	4
2	A	Guard	22	6
3	A	Guard	24	9
4	A	Forward	13	8
5	A	Forward	13	9
6	A	Guard	10	5
7	B	Guard	24	4
8	B	Guard	22	6
9	B	Forward	34	2
10	B	Forward	15	5
11	B	Forward	23	5
12	B	Guard	10	4

While printing the entire dataset is useful, when dealing with extremely large tables, efficiency demands a way to limit the output. The powerful `OBS=` option, specified within the data option parentheses, allows analysts to quickly review only the initial records. This is achieved by instructing **PROC PRINT** to process and output only the first 'N' number of rows, making it an indispensable tool for debugging and sanity checks without waiting for a full report generation. Here, we modify the syntax to output only the first five observations from `my_data`.

```
/*print first five rows of dataset*/
proc print data=my_data(obs=5);
```

Obs	team	position	points	assists
1	A	Guard	14	4
2	A	Guard	22	6
3	A	Guard	24	9
4	A	Forward	13	8
5	A	Forward	13	9

Method 3: Selecting Specific Variables (The VAR Statement)

Beyond controlling the number of rows displayed, analysts frequently need to restrict the columns, or variables, visible in the final report. This crucial step reduces clutter and ensures the report focuses only on relevant metrics. This capability is managed through the VAR statement, which is placed inside the **PROC PRINT** block.

The VAR statement allows the user to explicitly list the names of variables they wish to include in the output. A significant benefit of using this statement is the ability to dictate the precise order of columns in the generated report, regardless of their original sequence within the input dataset structure.

In the following example, we combine the row-limiting functionality (`OBS=5`) from the previous method with the column-selection capability of the VAR statement. The resulting output will contain only the `team` and `points` variables for the initial five records, demonstrating how to create concise and highly focused summary tables.

```
/*print rows for team and points variables only*/
proc print data=my_data(obs=5);
var team points;
run;
```

Obs	team	points
1	A	14
2	A	22
3	A	24
4	A	13
5	A	13

Method 4: Generating Grouped Reports (The BY and PROC SORT Statements)

For comparative analysis or when generating multi-section reports, it is often necessary to segment the output based on the unique values of a categorical variable. This powerful segmentation is achieved using the BY statement within the **PROC PRINT** procedure. However, there is a critical prerequisite: the input dataset must first be sorted according to the variable(s) specified in the BY statement.

This preliminary sorting step is handled efficiently by the PROC SORT procedure. If the data is not sorted beforehand, SAS will typically issue an error, as the BY statement relies on contiguous groups of identical values to correctly delineate report sections.

In the following code block, we first use PROC SORT to order the `my_data` dataset by the `team` variable. Subsequently, the **PROC PRINT** step utilizes the BY statement to instruct SAS to generate distinct report sections for each unique team. This mechanism significantly improves clarity when analyzing data segmented by key categories.

```
/*sort rows of dataset by values in team column*/
proc sort data=my_data;
by team;
run;

/*print entire dataset grouped by values in team column*/
proc print data=my_data;
by team;
run;
```

team=A

Obs	position	points	assists
1	Guard	14	4
2	Guard	22	6
3	Guard	24	9
4	Forward	13	8
5	Forward	13	9
6	Guard	10	5

team=B

Obs	position	points	assists
7	Guard	24	4
8	Guard	22	6
9	Forward	34	2
10	Forward	15	5
11	Forward	23	5
12	Guard	10	4

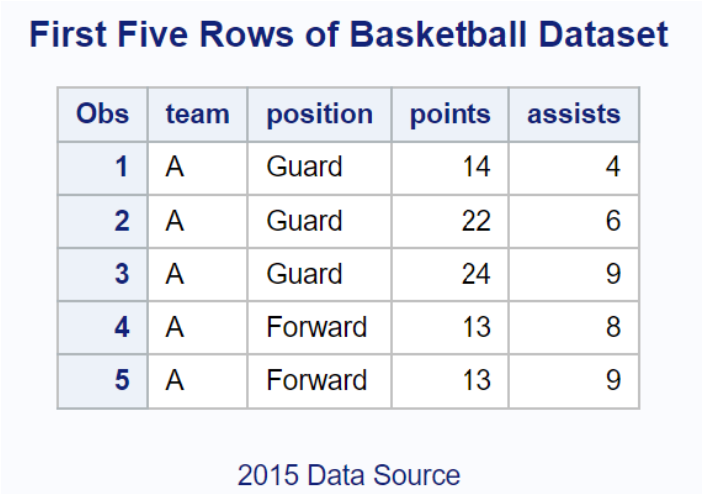
Method 5: Enhancing Presentation with Titles and Footers

In the context of formal reporting, adding context, traceability, and professionalism is paramount. **PROC PRINT** facilitates this easily through the use of the global `TITLE` and `FOOTNOTE` statements. These statements allow users to incorporate custom text that appears immediately above and below the output table, respectively, providing essential descriptive context for the report contents.

It is important to understand that `TITLE` and `FOOTNOTE` statements are considered global within a SAS session. Once defined, they remain active and affect the output of all subsequent procedures until they are explicitly cleared (e.g., using `TITLE;`) or overwritten by new statements. Therefore, careful placement before the target **PROC PRINT** step is necessary to ensure the context is applied correctly.

In this final example, we demonstrate how to define a specific title, "First Five Rows of Basketball Dataset," and attribute the source of the data using the `FOOTNOTE` statement, "2015 Data Source." Although these statements are placed before the **PROC SORT** step in the code block below, they take effect during the subsequent printing step, visibly enhancing the output's professionalism and traceability.

```
/*print dataset with title and footer*/
proc sort data=my_data;
title "First Five Rows of Basketball Dataset";
footnote "2015 Data Source";
run;
```



First Five Rows of Basketball Dataset

Obs	team	position	points	assists
1	A	Guard	14	4
2	A	Guard	22	6
3	A	Guard	24	9
4	A	Forward	13	8
5	A	Forward	13	9

2015 Data Source

Conclusion: Mastering Data Presentation in SAS

The PROC PRINT procedure is arguably the most versatile and fundamental utility for data visualization and basic report generation within the SAS programming ecosystem. Its simplicity for default output combined with its deep customization potential makes it a workflow staple.

By skillfully utilizing the core optional statements--specifically the `OBS` option for row limitation, the VAR statement for column selection, and the BY statement for grouping--users can efficiently transform raw dataset output into highly customized, readable, and informative reports that are perfectly tailored to specific analytical or organizational needs.

These five methods provide a robust, practical foundation for utilizing **PROC PRINT** effectively, enabling analysts to move quickly from data processing to polished data presentation.

Additional Resources

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