How to Use PROC PRINT in SAS (With Examples)

Authored by

stats writer

November 19, 2025

The PROC PRINT procedure is one of the most fundamental and frequently used tools within the SAS statistical software environment. It is specifically designed to display the contents of a SAS data set in a clean, tabular format, making it indispensable for data validation and quick review. Unlike procedures that perform complex statistical calculations, PROC PRINT focuses purely on visualization, allowing users to inspect the observations (rows) and variables (columns) stored in memory or on disk. This procedure is remarkably flexible, offering various options to customize the output, such as controlling the number of rows displayed, selecting specific variables, defining breaks for grouped data, and applying aesthetic enhancements like titles and footers. A solid understanding of PROC PRINT is essential for anyone working with SAS, as it serves as the primary method for confirming data integrity before moving onto advanced analysis.

Understanding the Core Syntax of PROC PRINT

At its simplest, running PROC PRINT only requires specifying the input data set using the DATA= option. This basic command generates a straightforward table listing all observations and all variables contained within the specified data set. The strength of this procedure, however, lies in its optional statements which allow for precise control over the output presentation. Mastering these options ensures that reports are not only accurate but also easily digestible by stakeholders.

To effectively utilize PROC PRINT, data analysts typically employ several methods, often combining statements like VAR, BY, and options like OBS= to tailor the resulting output table. Whether you need a full dump of a diagnostics data set or a highly summarized, polished report, PROC PRINT provides the necessary flexibility. The following sections detail five essential methods for generating customized reports in SAS using this robust procedure.

Five Essential Methods for Using PROC PRINT

When working with large or complex data sets, it is rarely efficient to display everything at once. We must learn to scope our view using specific statements. Below, we outline the fundamental approaches to generating tailored tabular reports using PROC PRINT.

Method 1: Print Entire Dataset

This is the default and simplest method, useful for quickly verifying small data sets or displaying final, complete tables.

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

Method 2: Print First N Observations in Dataset

To inspect a small sample of the data, the **OBS= option** is included directly within the DATA= statement parentheses. This is invaluable for debugging code or verifying data ingestion.

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

Method 3: Print Specific Variables in Dataset

The **VAR statement** specifies which columns (variables) should appear in the output. Omitting the VAR statement defaults to displaying all variables.

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

Method 4: Print Dataset Grouped by Specific Variable

When using the **BY statement**, PROC PRINT generates a separate section (or table) for each unique value of the specified BY variable. Note that the data must first be sorted by the BY variable using **PROC SORT**.

```
/*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;
```

Method 5: Print Dataset with Title and Footer

For report generation, adding contextual text via the **TITLE** and **FOOTNOTE** statements enhances clarity and professionalism. These global statements apply to the current procedure output.

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

```
run;
```

Preparing the Sample Data for Demonstration

To illustrate these five methods in practice, we will utilize a small sample data set named **my_data**. This data set contains crucial statistics related to basketball players, including their team assignment, position, and performance metrics like points and assists. This consistent data set will be used across all subsequent examples to clearly demonstrate the effect of each PROC PRINT option.

The following SAS DATA step code generates the necessary input table:

```
/*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;
```


This data set contains 12 observations (rows) and 4 variables (columns): `team`, `position`, `points`, and `assists`.

Example 1: Printing the Entire Dataset

As demonstrated in Method 1, the most basic use case involves displaying all records in the input table. This standard syntax is employed when a complete view of the data is required, often for final reporting or comprehensive data review.

```
/*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

Observe that the output table accurately reflects all 12 observations and all four variables present in the `my_data` data set.

Example 2: Limiting Output to the First N Observations

When working with massive data sets, displaying only the initial records can save significant processing time and make the output review faster. We achieve this by applying the **OBS= option** directly within the parentheses of the DATA= statement, restricting PROC PRINT to process only a specified number of rows.

```
/*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

As confirmed by the image above, the resulting output table displays precisely the first five observations from the `my_data` table, effectively demonstrating the filtering capability of the OBS= option.

Example 3: Selecting Specific Variables for Display

Often, a report requires only a subset of the available data attributes. The **VAR statement** allows the user to explicitly list the variables to be included in the output, filtering out any extraneous columns, thereby simplifying the generated report. Here, we restrict the view to only the `team` and `points` variables.

```
/*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

The resulting table confirms that only the selected **team** and **points** variables are displayed, streamlining the presentation of key performance indicators.

Example 4: Generating Grouped Reports using the BY Statement

For comparative analysis, it is often necessary to segment the output based on categorical data. The **BY statement** instructs PROC PRINT to produce separate reports for each unique value of the specified grouping variable (in this case, `team`). It is critical to use **PROC SORT** beforehand to ensure the data is properly ordered by the BY variable.

```
/*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

The resulting image clearly shows two distinct output sections: one detailed report for **Team A** and

a separate report for **Team B**, illustrating how the BY statement facilitates segregated reporting.

Example 5: Enhancing Reports with TITLE and FOOTNOTE Statements

To finalize a report for presentation, analysts must add metadata that clearly identifies the content, scope, and source of the data. The **TITLE** and **FOOTNOTE** global statements are used to place descriptive text at the top and bottom of the output page, respectively, adding context vital for interpretation.

```
/*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

As demonstrated, the title appears prominently above the data table, providing immediate context, while the footnote ensures proper citation and data source transparency below the output.

Conclusion and Further Learning

The PROC PRINT procedure is a powerful, yet straightforward, utility in SAS for generating clear, tabular summaries of data sets. By mastering the core syntax and optional statements like VAR, BY, and OBS=, users can quickly transform raw data into highly customized and informative reports suitable for both internal debugging and external presentation. These five methods represent the foundation of efficient data visualization within the SAS programming environment.

For analysts seeking to expand their reporting capabilities beyond basic printing, exploring related

procedures and techniques is highly recommended.

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