Learning Data Aggregation in SAS: A Guide to PROC MEANS with the CLASS Statement

Authored by

Mohammed looti

November 14, 2025

Mastering Grouped Statistical Analysis in SAS with PROC MEANS and the CLASS Statement

In the specialized domain of statistical programming and large-scale data analysis, the capacity to efficiently reduce massive, complex raw data into actionable, summarized figures is paramount.
The SAS System, widely recognized for its robust capabilities in advanced analytics and data management, provides powerful procedures specifically designed for this task.
The cornerstone of data summarization in SAS is the **PROC MEANS** procedure. This versatile tool calculates a comprehensive set of **summary statistics**--including measures of central tendency (mean), dispersion (standard deviation), and range (minimum, maximum)--for specified numeric variables within any given dataset. It is the essential first step in understanding data distribution and fundamental relationships.

While **PROC MEANS** excels at generating overall, global statistics, its true potential for comparative and detailed analysis is realized only through the implementation of the **CLASS statement**.
The introduction of this statement allows analysts to move beyond simple aggregation, enabling the computation of segmented statistics across distinct subgroups. These subgroups are typically defined by one or more **categorical variables**, providing the necessary foundation for deep, granular comparisons. This capability is absolutely vital for researchers and business professionals who must analyze performance variance, behavioral differences, or key trends across different predefined segments of their data.

This comprehensive tutorial is designed to provide practical, step-by-step guidance on leveraging the combined power of the **CLASS statement** within the **PROC MEANS** procedure in the SAS environment. We will use a realistic sample **dataset** focused on basketball player performance metrics. Through this practical scenario, we will demonstrate how to systematically segment and analyze crucial performance indicators (KPIs) based on criteria like team membership and player position, effectively transforming raw data into highly targeted, segmented intelligence.

Preparation: Constructing the Sample Dataset

Before initiating any statistical procedure in SAS, a properly structured **dataset** must be defined and loaded into memory. To effectively illustrate the grouping functionality of **PROC MEANS** when coupled with the **CLASS statement**, we will generate a small, foundational dataset named `my_data`. This dataset is explicitly designed to contain both variables necessary for classification and variables containing measurable performance metrics.

The classification variables, which will ultimately define our subgroups, are `team` and `position`. These are **categorical variables**. In contrast, the performance metrics--the subjects of our

statistical calculations--are the **numeric variables** `points` and `assists`. This blend of data types is essential for accurately showcasing the core functionality of any grouping procedure, highlighting the process of calculating numerical summaries based on qualitative categories.

The following SAS code block demonstrates the standard process for creating this sample data. We use the fundamental `DATA` step to define the structure of the dataset and the `DATALINES` statement to input the raw observations directly within the program script. Following the data creation, we execute **`PROC PRINT`**. This step serves as a crucial data verification check, allowing us to confirm that the dataset has been correctly loaded into the SAS environment and is structurally sound and ready for subsequent analytical procedures.

```
/*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;

/*view 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

Establishing a Baseline: Running PROC MEANS Without Grouping

Before we implement any segmentation, it is highly informative to examine the default operational behavior of **PROC MEANS** when it is executed without specifying any grouping criteria. This initial run provides a necessary statistical baseline, aggregating all available data points into a single, comprehensive summary block. Understanding this simple, non-grouped output is key to fully appreciating the added dimension and analytical power provided by incorporating the **CLASS statement** later in the analysis.

The following code block demonstrates the most fundamental application of **PROC MEANS**. By specifying only the input **dataset** (`my_data`), we instruct SAS to automatically scan the data, identify all **numeric variables** present, and calculate the default set of **summary statistics** (N, Mean, Std Dev, Min, Max) based on the entire population of observations simultaneously, without discrimination or segmentation.

```
/*calculate summary statistics for numeric variables*/
proc means data=my_data;
run;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
points	12	18.6666667	7.2780284	10.0000000	34.0000000
assists	12	5.5833333	2.1514618	2.0000000	9.0000000

The resulting output confirms that **PROC MEANS**, when run minimally, computes purely global **summary statistics**. For both the 'points' and 'assists' variables, we receive metrics such as N (the total count of 12 observations), the overall Mean, the Standard Deviation, and the Minimum and Maximum values across the entire dataset. While this aggregated view is helpful for determining the general magnitude and spread of the data, it critically fails to provide any insight into how these essential metrics vary between logical subgroups, such as the two different teams or the distinct player positions. This limitation is precisely why implementing classification variables is a necessity for any deeper, comparative data analysis.

Segmenting Data: Grouping by a Single Classification Variable

In nearly all professional analytical scenarios, global **summary statistics** are insufficient for decision-making. Analysts fundamentally require the ability to draw meaningful comparisons between predefined business or demographic segments. This essential task is executed through the **CLASS statement**, which transforms **PROC MEANS** from a simple aggregation tool into a powerful engine for comparative statistical reporting. By defining one or more grouping variables, we instruct SAS to compartmentalize the analysis, yielding entirely separate sets of statistics for each unique group detected.

For our first segmentation illustration, we will apply the **CLASS statement** using only the `team` variable. This instruction forces SAS to calculate the performance metrics (points and assists) independently for Team A and Team B. This basic segmentation is foundational for answering direct comparative questions, such as "Which team demonstrates a statistically higher average points per game?" or "Are there significant differences in the distribution of assists between the two teams?"

```
/*calculate summary statistics for numeric variables, grouped by team*/
proc means data=my_data;
class team;
run;
```

The MEANS Procedure

team	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
A	6	points	6	16.0000000	5.6213877	10.0000000	24.0000000
		assists	6	6.8333333	2.1369761	4.0000000	9.0000000
B	6	points	6	21.3333333	8.2381228	10.0000000	34.0000000
		assists	6	4.3333333	1.3662601	2.0000000	6.0000000

The output generated by this execution clearly demonstrates the effectiveness of the **CLASS statement**. Instead of one overall summary, we now receive two distinct blocks of **summary statistics**--one for each unique value found in the `team` classification column. The table first presents the statistics for 'Team A', detailing the mean points (17.83) and mean assists (6.83), alongside their standard deviations and range limits. Immediately following, a parallel and independent set of statistics is provided for 'Team B'. This grouped presentation facilitates immediate and straightforward comparison, revealing, for instance, that while Team B maintains a higher mean score in points (21.33), Team A appears to lead in average assists. This granular view is indispensable for comparative performance evaluation and highly targeted reporting.

Achieving Granular Insight: Grouping by Multiple Variables

The analytical power of the **CLASS statement** extends significantly beyond grouping by just a single variable. For addressing highly complex analytical questions, it is frequently necessary to segment the data across multiple dimensions simultaneously. By simply listing multiple **categorical variables** within the **CLASS statement**, we effectively create a hierarchical grouping structure. This results in the generation of highly specific summary statistics for every unique combination formed by the intersection of these variables.

In this next practical example, our goal is to understand player performance not merely by their assigned team, but specifically by the player position within that team. To achieve this essential fine-grained segmentation, we must modify our **PROC MEANS** code to include both `team` and `position` in the **CLASS statement**. The sequence in which these variables are listed dictates the hierarchy of the resulting output table; in this case, the procedure will first group the data by team, and then proceed to subgroup by position within each respective team.

```
/*calculate summary statistics for numeric variables, grouped by team and position*/
proc means data=my_data;
class team position;
run;
```

The MEANS Procedure

team	position	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
A	Forward	2	points	2	13.0000000	0	13.0000000	13.0000000
			assists	2	8.5000000	0.7071068	8.0000000	9.0000000
	Guard	4	points	4	17.5000000	6.6080759	10.0000000	24.0000000
			assists	4	6.0000000	2.1602469	4.0000000	9.0000000
B	Forward	3	points	3	24.0000000	9.5393920	15.0000000	34.0000000
			assists	3	4.0000000	1.7320508	2.0000000	5.0000000
	Guard	3	points	3	18.6666667	7.5718778	10.0000000	24.0000000
			assists	3	4.6666667	1.1547005	4.0000000	6.0000000

A careful observation of the output table reveals a substantially deeper and more detailed analysis. The **summary statistics** for the **numeric variables** (points and assists) are now carefully segmented across four highly distinct groups: 'Team A - Forward', 'Team A - Guard', 'Team B - Forward', and 'Team B - Guard'. This precise level of segmentation permits analysts to identify critical nuances, such as whether the overall scoring difference between the two teams is primarily driven by their forwards or, conversely, by the performance of their guards.

For example, we can now directly compare the average points scored by Team A Guards (17.50) versus Team B Guards (18.00). This critical level of detail is necessary for strategic decision-making, allowing users to move far beyond general, high-level averages and pinpoint performance metrics within very specific operational segments of the **dataset**. This multi-variable grouping capability stands as a cornerstone of sophisticated data exploration and reporting within the SAS platform.

Enhancing Control: Advanced Options and Best Practices

Although the basic implementation of **PROC MEANS** alongside the **CLASS statement** is immensely powerful, SAS provides several supplementary statements that significantly enhance control and flexibility over the statistical output, allowing analysts to tailor the results precisely to their reporting needs.

One of the most frequently used refining statements is the **`VAR`** statement, which is utilized to explicitly select which **numeric variables** should be the subject of the analysis. If the `VAR` statement is omitted, **PROC MEANS** defaults to summarizing every numeric column found in the input data. Conversely, the **`OUTPUT`** statement is essential for the practical operationalization of results; it grants the analyst the ability to save the computed **statistics** (such as the calculated group means and standard deviations) into a brand new SAS **dataset**. This newly created dataset can then be seamlessly used as input for subsequent statistical modeling, advanced reporting, or

for merging with other data sources.

For users managing complex, multi-variable classifications, the **`TYPES`** and **`WAYS`** options offer surgical control over the aggregation hierarchy. The **`TYPES`** statement specifically defines which combinations of the CLASS variables should be summarized. For instance, if you classify variables A, B, and C, you might use `TYPES` to request only summaries for the combination (A B) and the full combination (A B C), while purposefully excluding summaries for A, B, C, or (A C). Similarly, the **`WAYS`** statement allows the user to specify the exact number of classification variables to be used in the grouping (e.g., `WAYS 1` requests all single-variable summaries, while `WAYS 2` requests all possible two-variable combinations). Utilizing these advanced options ensures that the analysis remains focused and avoids the generation of unnecessary or statistically redundant output tables, streamlining the analytical workflow.

Conclusion

The efficient and seamless integration of the **PROC MEANS** procedure and the **CLASS statement** represents one of the most fundamental and demonstrably powerful techniques available for robust data analysis within the SAS programming environment. This combination moves far beyond simple overall data aggregation by specifically enabling analysts to conduct reliable comparative studies across predefined cohorts or categories.

By facilitating the precise calculation of **summary statistics** segmented by one or more **categorical variables**, the **CLASS statement** grants the analyst the power to uncover specific trends, accurately identify performance variances, and rigorously validate hypotheses that would otherwise remain hidden when viewing only the general, non-grouped data average. Whether your task involves conducting detailed A/B testing, evaluating demographic differences in survey results, or assessing performance metrics across distinct operational units, mastering the grouped statistical capabilities of **PROC MEANS** is absolutely essential for any professional seeking to extract deeper, more actionable intelligence from their data.

Further Resources and Learning

To further refine your expertise in the powerful SAS programming environment, we highly recommend exploring these related tutorials and official documentation links:

Reviewing the official SAS documentation for the `VAR` statement within PROC MEANS to control variable selection.
Learning the essential use of the `OUTPUT` statement to export statistical results into new SAS datasets for further use.
Exploring the functional differences between **PROC MEANS**, **PROC SUMMARY**, and **PROC TABULATE** for various complex reporting needs.