Learning Data Comparison with SAS: A Guide to Using PROC COMPARE

Authored by

Mohammed looti

October 30, 2025

In modern data analysis, maintaining the consistency and integrity of information is paramount. The ability to quickly and accurately identify differences and similarities between datasets is essential for ensuring robust data quality and validating complex analytical processes. Within the powerful environment of SAS, the **PROC COMPARE** procedure stands out as an indispensable utility designed specifically for this critical task. It allows users to efficiently pinpoint discrepancies--or confirm exact matches--between two SAS datasets.

Whether you are performing routine data auditing, validating the output of a data transformation pipeline, or reconciling different versions of historical data, **PROC COMPARE** provides a comprehensive and meticulously detailed report. This procedure is the cornerstone of effective change management for data assets, ensuring that every modification, update, or migration maintains consistency and reliability across all stages of your workflow.

Mastering the Essential Syntax of PROC COMPARE

The fundamental structure for executing the **PROC COMPARE** procedure is remarkably clear and intuitive. To initiate a comparison, the procedure requires the specification of two primary components: the base dataset, which serves as the reference standard, and the comparison dataset, which is measured against the base. The procedure then systematically analyzes these two inputs, highlighting any structural or value-based discrepancies.

The core syntax for invoking this powerful tool is defined by the PROC statement and its required options, as shown below:

```
proc compare
base=data1
compare=data2;
run;
```

In this structure, the option **base=data1** explicitly designates the primary reference dataset against which all measurements will be taken. Conversely, **compare=data2** specifies the dataset whose contents and structure will be evaluated relative to the base. Once executed, the procedure generates a systematic, multi-part report that outlines all differences identified between the two data structures.

A Practical Example: Preparing Datasets for Comparison

To fully appreciate the utility of **PROC COMPARE**, let us walk through a practical scenario involving two slightly different datasets. We will use the SAS DATA step to construct simple sample data structures--named `data1` and `data2`--that will serve as our base and comparison inputs,

respectively. This setup is crucial for demonstrating how the procedure handles varying structures and contents in a controlled environment.

Our two datasets contain information about hypothetical sports teams. Notice that `data1` includes an additional statistical field, and both datasets contain different entries and values for some teams. These deliberate differences will allow **PROC COMPARE** to showcase its capabilities in detail. We utilize the familiar `DATA` step and `DATALINES` statement, a standard method for direct data input in SAS demonstrations:

```
/*Create the sample datasets*/
data data1;
input team $ points rebounds;
datalines;
A 25 10
B 18 4
C 18 7
D 24 12
E 27 11
;
run;

data data2;
input team $ points;
datalines;
A 25
B 18
F 27
G 21
H 20
;
run;

/*View the datasets using PROC PRINT*/
proc print data=data1;
proc print data=data2;
```

Executing the code above, along with the subsequent `PROC PRINT` statements, allows us to visually verify that the data has been loaded correctly into the SAS environment. This preliminary step of inspecting the base data is essential for confirming the starting point before initiating the comparison procedure.

Obs	team	points	rebounds
1	A	25	10
2	B	18	4
3	C	18	7
4	D	24	12
5	E	27	11

Obs	team	points
1	A	25
2	B	18
3	F	27
4	G	21
5	H	20

As illustrated by the output, `data1` possesses three variables: `team`, `points`, and `rebounds`, whereas `data2` only contains `team` and `points`. Furthermore, a closer inspection reveals differences in the specific observations (rows) recorded for several teams. These inherent structural and content variations provide a rich example for demonstrating the analytical power of **PROC COMPARE**.

Executing the Comparison and Deciphering the Output

With both the base (`data1`) and comparison (`data2`) datasets now prepared, we proceed to execute the core **PROC COMPARE** statement. This action initiates a thorough analysis of the two data structures, resulting in a comprehensive report typically divided into three distinct tables. Each table offers a unique layer of insight into the similarities and, more importantly, the differences between the two inputs.

The code required to launch the comparison is simple and direct:

```
/*Compare the two datasets: data1 (base) vs data2 (compare)*/
proc compare
base=data1
compare=data2;
run;
```

Upon execution, **PROC COMPARE** generates a sequence of detailed outputs. These reports are meticulously structured to guide the user from a high-level overview of structural mismatches down to the granular, row-by-row differences in values. Understanding the content of each of these output tables is paramount to effective data validation.

Interpreting Output Table 1: Dataset Summary and Structure

The initial report produced by **PROC COMPARE** serves as a high-level structural diagnostic. It provides a quick, essential overview of both the base and comparison datasets, immediately highlighting potential mismatches in dimensionality or composition before diving into specific value differences. This summary acts as an immediate check on the fundamental alignment of the two data sources.

```
                    The COMPARE Procedure
           Comparison of WORK.DATA1 with WORK.DATA2
                        (Method=EXACT)

                      Data Set Summary

Dataset               Created          Modified  NVar    NObs

WORK.DATA1   07MAR22:14:31:17  07MAR22:14:31:17     3       5
WORK.DATA2   07MAR22:14:31:17  07MAR22:14:31:17     2       5


                      Variables Summary

Number of Variables in Common: 2.
Number of Variables in WORK.DATA1 but not in WORK.DATA2: 1.
```

Key information provided in this table includes the counts of variables and observations in each dataset, as well as an accounting of which variables are unique or shared:

Dimensionality Metrics: The table lists the number of variables (NVar) and observations (NObs) found in each dataset. For our example, the metrics are:

`data1` contains 3 variables and 5 observations.
`data2` contains 2 variables and 5 observations.

This immediately confirms a structural difference: `data1` contains the variable 'rebounds' which is missing in `data2`.

Common Variable Count: The report clearly identifies the number of variables shared by both

datasets. In our case, `data1` and `data2` share 2 variables: 'team' and 'points'. This section defines the scope of the value comparison that follows, ensuring that the user knows exactly which fields are being analyzed for content discrepancies.

Interpreting Output Table 2: Quantification of Value Differences

While the first table addresses structure, the second output generated by **PROC COMPARE** focuses purely on content, providing a concise quantitative summary of the value differences observed across the shared variables. This table is essential for quantifying the extent of data discrepancies and assessing the overall health of the comparison datasets.

```
                    Observation Summary

                Observation      Base  Compare

                First Obs           1        1
                First Unequal       3        3
                Last  Unequal      5        5
                Last  Obs           5        5

Number of Observations in Common: 5.
Total Number of Observations Read from WORK.DATA1: 5.
Total Number of Observations Read from WORK.DATA2: 5.

Number of Observations with Some Compared Variables Unequal: 3.
Number of Observations with All Compared Variables Equal: 2.


                  Values Comparison Summary

Number of Variables Compared with All Observations Equal: 0.
Number of Variables Compared with Some Observations Unequal: 2.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 9.


           All Variables Compared have Unequal Values

              Variable  Type  Len  Ndif   MaxDif

              team      CHAR    8     3
              points    NUM     8     3    9.000
```

This report summarizes, variable by variable, the number of observations where the values do not match between the base and comparison inputs. The aggregate summary provided at the bottom of the table is particularly insightful:

For the **team** variable, the report indicates 3 observations contain different values. This means that for three corresponding rows, the team identifier in `data1` does not equal the team identifier in `data2`.

For the **points** variable, 3 observations also exhibit differing values. Crucially, for numeric variables, **PROC COMPARE** also calculates and reports the "Max Difference," which is 9 in this instance. This figure represents the largest absolute magnitude of difference found between any corresponding 'points' value in the two datasets. This metric is invaluable for quickly judging the severity of numerical discrepancies, guiding subsequent data auditing steps.

Interpreting Output Table 3: Granular Row-by-Row Differences

The third and most detailed output table from **PROC COMPARE** presents the actual, granular differences found between observations, displaying the exact conflicting values side-by-side. This level of detail is indispensable for precise data validation and debugging processes.

The COMPARE Procedure
Comparison of WORK.DATA1 with WORK.DATA2
(Method=EXACT)

Value Comparison Results for Variables

Obs	Base Value team	Compare Value team
3	C	F
4	D	G
5	E	H

Obs	Base points	Compare points	Diff.	% Diff
3	18.0000	27.0000	9.0000	50.0000
4	24.0000	21.0000	-3.0000	-12.5000
5	27.0000	20.0000	-7.0000	-25.9259

The report segments the output by variable, allowing users to see the specific data points causing the mismatch:

For the **team** variable, the table clearly shows the specific location of discrepancies. For instance,

focusing on the third observation, `data1` holds a 'team' value of **C**, while `data2` holds **F**. This direct visual comparison instantly identifies data entry errors or changes in key identifier fields.
Similarly, for the **points** variable, the output highlights numerical value deviations. In the third observation, `data1` records **18** points, compared to **27** points in `data2`. The calculated difference of **9** is also explicitly listed, confirming the maximum difference identified in the second summary table. This granular comparison is critical for verifying numerical calculations and debugging ETL pipelines.

Collectively, these three tables provide a complete, nuanced picture of the relationship between the two datasets, empowering users to identify, diagnose, and resolve all forms of data inconsistencies effectively.

Focusing the Analysis: Comparing Specific Variables

While a comprehensive comparison of all shared variables is often necessary, efficiency sometimes dictates focusing the analysis on a targeted subset of fields. **PROC COMPARE** facilitates this targeted approach through the use of the **VAR** statement. This allows data professionals to narrow the scope of the comparison, significantly reducing report clutter and streamlining the analysis of highly specific changes.

If, for instance, the objective is only to compare the 'points' variable between `data1` and `data2`, the standard **PROC COMPARE** statement is augmented by the **VAR** option, followed by the name of the variable(s) of interest:

```
/*Compare the differences between the datasets only for 'points' variable*/
proc compare
base=data1
compare=data2;
var points;
run;
```

Executing this modified code yields the same three types of output tables, but the analysis and reporting of differences are restricted solely to the 'points' variable. This targeted methodology is invaluable for performing quick, iterative data validation checks or when specific numerical consistency is the only concern.

Advanced Options for Robust Data Reconciliation

Beyond its fundamental comparison capabilities, **PROC COMPARE** is equipped with a rich set of advanced options that allow for customized behavior and detailed output generation, adapting it to

nearly any complex data analysis or reconciliation requirement. Utilizing these options can significantly enhance your capacity for rigorous data auditing.

A selection of the most powerful advanced options includes:

ALLSTATS: This option forces the procedure to output comprehensive descriptive statistics (e.g., minimum, maximum, mean, standard deviation) for all matched numeric variables, providing insight into statistical consistency, not just value equality.
LISTALL: Typically, **PROC COMPARE** only reports differences. Specifying LISTALL ensures that all variables and their values are listed in the output, including those that are identical between the base and comparison datasets, offering a complete side-by-side view.
OUT=*output-dataset*: This critical option allows the user to save the results of the comparison--specifically the identified differences--into a new SAS dataset. This is essential for programmatic error handling, reporting, or subsequent automated analysis.
ID=*variable(s)*: By default, the procedure matches observations based on their physical order (row number). The ID statement overrides this, enabling the procedure to match observations based on the values of specified key identifier variables. This is vital when datasets are not sorted identically or when records might be missing.
CRITERION=*value*: Highly valuable for numeric comparisons, this option sets a tolerance level for equality. For instance, setting CRITERION=0.0001 dictates that two numerical values differing by less than this tolerance are considered equal. This addresses common issues related to floating-point arithmetic inaccuracies.

When incorporating these advanced features, it is always considered a best practice to consult the official SAS documentation for **PROC COMPARE**. A thorough understanding of these options is key to leveraging the full potential of the procedure for maintaining high data quality standards.

Conclusion: Securing Data Integrity with PROC COMPARE

The **PROC COMPARE** procedure is a vital component of the SAS toolkit, serving as the primary mechanism for establishing and maintaining data quality across complex analytical projects. Its robust ability to swiftly identify and document differences between datasets--covering structural variations, metadata discrepancies, and granular value changes--makes it an essential tool for data governance.

By mastering its straightforward syntax, understanding its three detailed output tables, and utilizing its advanced options, data professionals can confidently verify changes, audit ETL processes, and ensure the absolute integrity of their data assets. For ongoing expertise and technical reference, the comprehensive SAS documentation remains the definitive source for maximizing the utility of **PROC COMPARE**.

Additional Resources for SAS Proficiency

To further advance your expertise in SAS and enhance your overall proficiency in data analysis, we recommend exploring the following related tutorials that cover other fundamental procedures and functionalities:

How to Use PROC SORT in SAS
Understanding the SAS DATA Step
Merging Datasets in SAS
Performing Frequency Analysis with PROC FREQ

These resources offer practical insights into managing, manipulating, and reporting data within the SAS environment, providing a solid foundation that complements the capabilities of powerful procedures like **PROC COMPARE**.