MWSUG 2025 – Paper PL-014

Applications of PROC COMPARE to Parallel Programming and other projects

Jay Iyengar, Data Systems Consultants LLC

ABSTRACT

PROC COMPARE is a valuable BASE SAS® procedure which is used heavily in the Pharma industry and other areas. By default, the capability of PROC COMPARE is to reconcile two data sets to determine if they have equivalent sets of records and sets of variables. In the clinical field and elsewhere, PROC COMPARE is often used to validate data sets in projects which involve parallel programming, where programmers independently perform the same tasks. In this paper, I will discuss the role PROC COMPARE plays in different SAS tasks, including DATA STEP merges, parallel programming, generation data sets, and more.

INTRODUCTION

PROC COMPARE has extensive capabilities that can be effectively utilized for the purposes of data validation and data quality. In its simplest usage, PROC COMPARE compares two SAS data sets which are defined as the BASE data set and the COMPARE data set. By default, PROC COMPARE compares data set attributes, variables, variable attributes, observations, and data values. PROC COMPARE generates multiple sections of output with a different section of output corresponding to a specific type of comparison. Data set attributes include the number of variables and number of observations in the data set and the date and time when the SAS data set was created and modified. Variable attributes include variable type (numeric vs. character), format, informat, length and label.

SAS DATA SETS USED IN PAPER EXAMPLES

For the first few examples in this paper which illustrate the features of PROC COMPARE, SAS data sets are used from the SASHELP library in SAS Studio. Two specific SAS datasets, SASHELP.PRDSAL2 and SASHELP.PRDSAL3 are used for the examples. The data sets contain actual and predicted sales figures for different types of furniture as well as other demographic and date/time variables.

PROC COMPARE BASICS

PROC COMPARE is limited to comparing two SAS data sets at a time. These two data sets are known as the BASE data set and the COMPARE data set. With the basic syntax, PROC COMPARE compares all the variables contained in both data sets. The basic syntax for PROC COMPARE is provided in Figure 1.

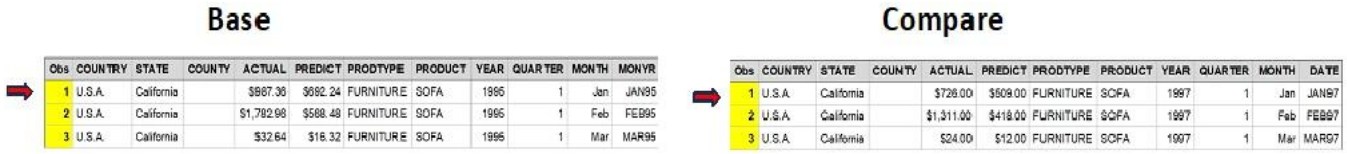
```
PROC COMPARE BASE=SAS-data-set  COMPARE=SAS-data-set <options>;
RUN;
```

Figure 1. PROC COMPARE basic syntax

How does PROC COMPARE actually execute a comparison? That is, what are the mechanics of PROC COMPARE. It compares observations by position. Position in a SAS data set is defined by observation number. It compares observations which have matching observation numbers.

The following set of diagrams and figures illustrate the mechanics of the process SAS uses under the hood to compare data sets using PROC COMPARE.

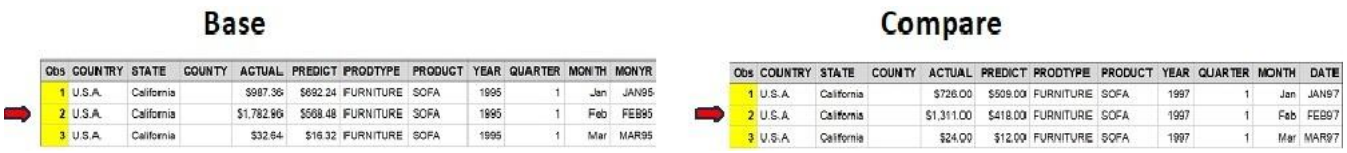
SAS reads the first observation from the BASE data set and compares that with the first observation from the COMPARE data set.



Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	MONYR
1	U.S.A.	California		$987.36	$692.24	FURNITURE	SOFA	1995	1	Jan	JAN95
2	U.S.A.	California		$1,782.96	$568.48	FURNITURE	SOFA	1995	1	Feb	FEB95
3	U.S.A.	California		$32.64	$16.32	FURNITURE	SOFA	1995	1	Mar	MAR95

Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	DATE
1	U.S.A.	California		$726.00	$509.00	FURNITURE	SOFA	1997	1	Jan	JAN97
2	U.S.A.	California		$1,311.00	$418.00	FURNITURE	SOFA	1997	1	Feb	FEB97
3	U.S.A.	California		$24.00	$12.00	FURNITURE	SOFA	1997	1	Mar	MAR97

It moves down to the second observation from the BASE data set and reads and compares that with the second observation from the COMPARE data set.



Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	MONYR
1	U.S.A.	California		$987.36	$692.24	FURNITURE	SOFA	1995	1	Jan	JAN95
2	U.S.A.	California		$1,782.96	$568.48	FURNITURE	SOFA	1995	1	Feb	FEB95
3	U.S.A.	California		$32.64	$16.32	FURNITURE	SOFA	1995	1	Mar	MAR95

Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	DATE
1	U.S.A.	California		$726.00	$509.00	FURNITURE	SOFA	1997	1	Jan	JAN97
2	U.S.A.	California		$1,311.00	$418.00	FURNITURE	SOFA	1997	1	Feb	FEB97
3	U.S.A.	California		$24.00	$12.00	FURNITURE	SOFA	1997	1	Mar	MAR97

Proceeding sequentially, it reads the third observation from the BASE data set and compares that with the third observation from the COMPARE data set.

Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	MONYR
1	U.S.A.	California		$987.36	$692.24	FURNITURE	SOFA	1995	1	Jan	JAN95
2	U.S.A.	California		$1,782.96	$568.48	FURNITURE	SOFA	1995	1	Feb	FEB95
3	U.S.A.	California		$32.64	$16.32	FURNITURE	SOFA	1995	1	Mar	MAR95

Obs	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	DATE
1	U.S.A.	California		$726.00	$509.00	FURNITURE	SOFA	1997	1	Jan	JAN97
2	U.S.A.	California		$1,311.00	$418.00	FURNITURE	SOFA	1997	1	Feb	FEB97
3	U.S.A.	California		$24.00	$12.00	FURNITURE	SOFA	1997	1	Mar	MAR97

This process continues moving sequentially from observation to observation until all matching observations have been read and compared from both data sets.

COMPARING VARIABLES IN DIFFERENT DATA SETS

The first example is the most basic example of PROC COMPARE and demonstrates its usage in comparing the data and attributes of two distinct SAS data sets. By default, PROC COMPARE will produce output comparing all the variables in both data sets, and comparing all the observations in both data sets. The SAS code for example 1 is provided below.

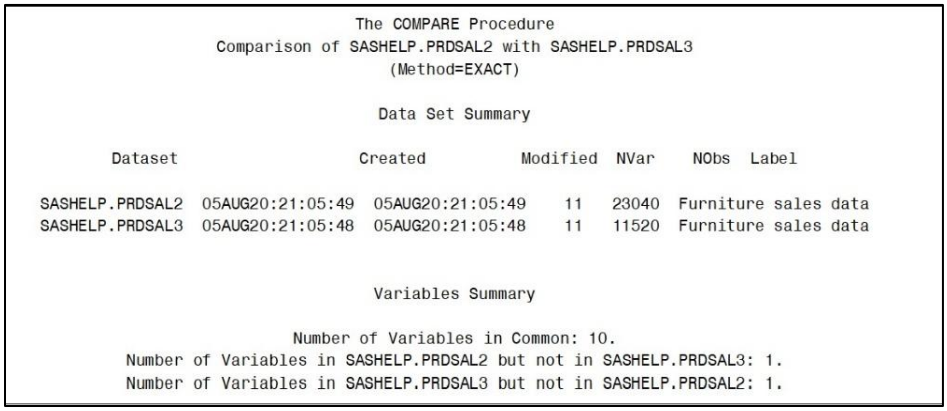
```
Proc Compare Base=SASHELP.PRDSAL2 Compare=SASHELP.PRDSAL3;
Run;
```

As shown in the code, the SAS data sets SASHELP.PRDSAL2 and SASHELP.PRDSAL3 were compared, with the PRDSAL2 as the BASE data set and PRDSAL3 as the COMPARE data set.

In this basic example, PROC COMPARE will produce multiple sections of output, which is the action it performs by default. These sections include the data set summary, observations summary, values comparison summary, variables with unequal values, and values comparison results for variables.

THE DATA SET SUMMARY

The first section of output is the data set summary. In the data set summary, the number of variables, number of observations, and the data set labels are compared. Also, in this section is the variables summary. The variables summary provides the number of variables the data sets have in common, as well as the number of variables which are unique to one data set or the other.



The COMPARE Procedure
Comparison of SASHELP.PRDSAL2 with SASHELP.PRDSAL3
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs	Label
SASHELP.PRDSAL2	05AUG20:21:05:49	05AUG20:21:05:49	11	23040	Furniture sales data
SASHELP.PRDSAL3	05AUG20:21:05:48	05AUG20:21:05:48	11	11520	Furniture sales data

Variables Summary

Number of Variables in Common: 10.
Number of Variables in SASHELP.PRDSAL2 but not in SASHELP.PRDSAL3: 1.
Number of Variables in SASHELP.PRDSAL3 but not in SASHELP.PRDSAL2: 1.

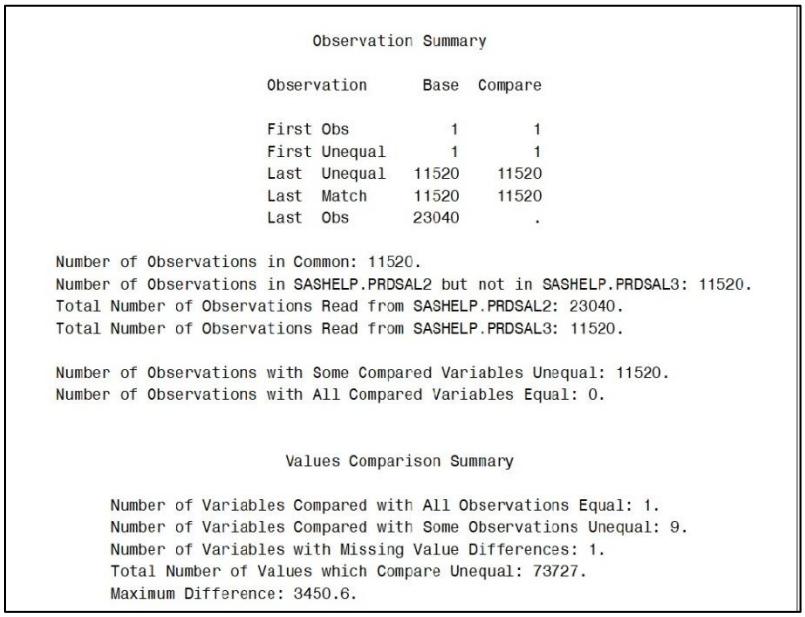
Figure 2. Data Set Summary

As you can see in Figure 2, the data set summary shows that both data sets contain 11 variables. However, PRDSAL2 has a larger number of observations than PRDSAL3; 23040 to 11520, respectively. This is a substantial difference in observations. The other information in the data set summary, date created, data modified, and data set label are the same for both data sets.

The variables summary shows that the data sets had 10 variables in common. Therefore, each data set had a single variable exclusive to that data set. Thus, there's one variable on PRDSAL2 not found on PRDSAL3. Likewise, there's one variable on PRDSAL3, which was not found on PRDSAL2.

THE OBSERVATION SUMMARY

The next section of output is the Observation Summary. The observation summary contains the results from comparing the observations, including how many observations the data sets had in common, and how many observations were found on one data set, but not the other, and vice versa. PRDSAL2 contains exactly twice the number of observations as PRDSAL3.



Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	11520	11520
Last Match	11520	11520
Last Obs	23040	.

Number of Observations in Common: 11520.
Number of Observations in SASHELP.PRDSAL2 but not in SASHELP.PRDSAL3: 11520.
Total Number of Observations Read from SASHELP.PRDSAL2: 23040.
Total Number of Observations Read from SASHELP.PRDSAL3: 11520.

Number of Observations with Some Compared Variables Unequal: 11520.
Number of Observations with All Compared Variables Equal: 0.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 9.
Number of Variables with Missing Value Differences: 1.
Total Number of Values which Compare Unequal: 73727.
Maximum Difference: 3450.6.

Figure 3. Observation Summary

It's interesting to point out the number of observations in common does not mean that the variable values for those observations were found equal. It just means that the data sets contained the same set of numbered observations. Although this is different from the total number of observations.

As shown in Figure 3, the observation summary indicates that 11520 observations were found in both data sets, and that the 11520 observations (the same number) were found in PRDSAL2, but not in PRDSAL3. It then provides the number of observations which were read from both data sets.

The observation summary also includes the number of observations where some variables were unequal, as well as the number of observations where all variables were equal. 11520 observations had some variables which were unequal, which is the result of comparing variable values for those observations. Similarly, no observations had all compared variables equal.

For PRDSAL2, the first 11520 observations all contained some variable values which were not equal to the variables in the 11520 observations in PRDSAL3. The last 11520 observations in PRDSAL2 could not be compared to observations in PRDSAL3, because they didn't exist in that SAS data set.

Below the observation summary is a values comparison summary which provides the number of variables with all observations equal, some observations unequal, missing value differences, and other statistics.

VARIABLES WITH UNEQUAL VALUES

The section of the output provides information on each variable on the data sets which have unequal values. For each variable, PROC COMPARE provides the **NDIF** – the number of records which have differences, **MAXDIF**, the maximum difference between variable values, and **MISSDIF**, the number of records which have differences which are due to missing values.

The COMPARE Procedure
Comparison of SASHELP.PRDSAL2 with SASHELP.PRDSAL3
(Method=EXACT)

Variables with Unequal Values

Variable	Type	Len	Label	Ndif	MaxDif	MissDif
COUNTRY	CHAR	10	Country	4608		0
STATE	CHAR	22	State/Province	9792		0
COUNTY	CHAR	20	County	7488		6336
ACTUAL	NUM	8	Actual Sales	11520	3417	0
PREDICT	NUM	8	Predicted Sales	11519	3451	0
PRODTYPE	CHAR	10	Product Type	5760		0
PRODUCT	CHAR	10	Product	8640		0
YEAR	NUM	8	Year	8640	3.000	0
MONTH	NUM	8	Month	5760	365	0

Figure 4. Variables with Unequal Values

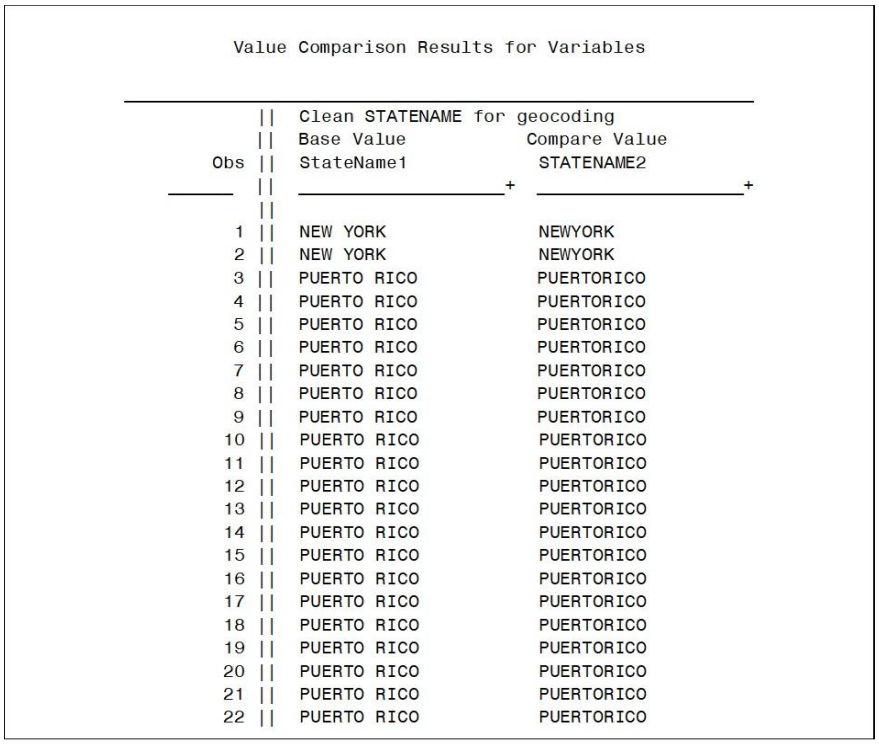
Figure 4 shows that the numeric variables ACTUAL and PREDICT have the largest number of records with differences, at 11520 and 11519 respectively.

Only 4 variables display the maximum difference. The 4 variables are numeric variables, because this measure isn't applicable to character variables.

The variable PREDICT has the largest maximum difference at 3451. COUNTY has the highest number of records with differences due to missing values. There are 6336 records in COUNTY which have differences due to missing values.

VALUES COMPARISON RESULTS

The values comparison results for variables section of output is a detailed report listing records with discrepant data values for specific variables.



Value Comparison Results for Variables

Obs	Clean STATENAME for geocoding Base Value StateName1	Compare Value STATENAME2
1	NEW YORK	NEWYORK
2	NEW YORK	NEWYORK
3	PUERTO RICO	PUERTORICO
4	PUERTO RICO	PUERTORICO
5	PUERTO RICO	PUERTORICO
6	PUERTO RICO	PUERTORICO
7	PUERTO RICO	PUERTORICO
8	PUERTO RICO	PUERTORICO
9	PUERTO RICO	PUERTORICO
10	PUERTO RICO	PUERTORICO
11	PUERTO RICO	PUERTORICO
12	PUERTO RICO	PUERTORICO
13	PUERTO RICO	PUERTORICO
14	PUERTO RICO	PUERTORICO
15	PUERTO RICO	PUERTORICO
16	PUERTO RICO	PUERTORICO
17	PUERTO RICO	PUERTORICO
18	PUERTO RICO	PUERTORICO
19	PUERTO RICO	PUERTORICO
20	PUERTO RICO	PUERTORICO
21	PUERTO RICO	PUERTORICO
22	PUERTO RICO	PUERTORICO

Figure 5. Value Comparison Results for Variables

In Figure 5 above, is the value comparison results section comparing the variables STATENAME1 and STATENAME2. This section of the output displays records in the data sets where data values for STATENAME1 and STATENAME2 are not equal.

As shown in Figure 5, the records which had differences were cases where the state name was two words, and STATENAME1 had an embedded space between words in the data value, whereas STATENAME2 concatenated the two words without a space.

COMPARING SPECIFIC VARIABLES IN DATA SETS

The BASE and COMPARE data sets are usually different SAS data sets which need to be reconciled with one another. However, instead of two data sets, PROC COMPARE can compare multiple variables within a single SAS data set. For the first example, we'll compare two different SAS data sets, PRDSAL2 and PRDSAL3.

PROC COMPARE gives you the ability to limit the variables in your comparison and thus specify individual variables in the data sets which you want to compare. This is often a useful feature of PROC COMPARE since if a data set contains many variables the PROC COMPARE output from a comparison of the data set will be of a very large volume.

The variables which you select can be in different data sets, or within the same data set. To specify the variables to compare, you use the VAR and WITH statements. When the variables are in different data sets, the VAR statement specifies the variable from the BASE data set, and the WITH statement the

variable from the COMPARE data set. However, if the variables are both in the same data set, then both statements specify variables from the BASE data set.

VARIABLES IN DIFFERENT DATA SETS

There are several reasons why as a programmer you'd want to compare individual variables contained in separate data sets. It might be the case that you need to merge the data sets and the variables which you compare are BY variables which are used to join the data sets. It also might be the case that you have multiple versions of the same SAS data set, and you need to reconcile the data sets with one another. Some of the specific cases will be explored in later sections of the paper.

In the first example, we're working with SAS data sets containing health care claims. The two data sets are HLTHDAT.ip2010claim, a header claim file, and HLTHDAT.ip2010line, a detail claim file. We're interested in comparing the claim identifier variable, CLM_ID, located on both data sets. The SAS code is presented below.

```
Proc Compare Base = HLTHDAT.ip2010claim Compare=HLTHDAT.ip2010line;
    Var CLM_ID;
    With CLM_ID;
Run;
```

VARIABLES WITHIN THE SAME DATA SET

When you need to compare specific variables contained within the same data set, you can use some of the same syntax as the separate data sets example. Since there's only one data set, you only use the BASE= to specify the data set, as there is no COMPARE= data set. You use the VAR and the WITH statements to reference the variables you're comparing.

In this example, we're working with the SAS data set, ZIPCODE. ZIPCODE contains multiple variables which record geographic information such as state. Specifically, there are 2 variables which contain this information, STATENAME1 and STATENAME2. These variables have been specified in the VAR and WITH statements, respectively.

The SAS code is presented below.

```
Proc Compare Base=ZIPCODE;
    Var StateName1;
    With StateName2;
Run;
```

For this example, the SAS Output has not been included because the volume of output is on a large scale. There are PROC COMPARE options which provide the ability to control the amount of output produced, through limiting the number of observations and other ways.

PROC COMPARE OPTIONS

The output produced from PROC COMPARE and what's contained in the output can be modified using options on the PROC COMPARE statement. A complete list of PROC COMPARE statement options and their descriptions are provided in Figure 6 below.

Option	Description
ALLOBS	includes the values for all matching observations.
ALLSTATS	prints a table of summary statistics for all pairs of matching variables.
ALLVARS	includes in the report the values and differences for all matching variables.
BRIEFSUMMARY	prints only a short comparison summary.
MAXPRINT=	restricts the number of differences to be printed.
NOSUMMARY	suppresses the data set, variable, observation, and values comparison summary reports.
NOVALUES	suppresses the report of the value comparison results.
PRINTALL	invokes the following options: ALLVARS, ALLOBS, ALLSTATS, LISTALL, and WARNING.
STATS	prints a table of summary statistics for all pairs of matching numeric variables that are judged unequal.
TRANSPOSE	prints the reports of value differences by observation instead of by variable.
LISTALL	lists all variables and observations that are found in only one data set.
LISTBASE	lists all observations and variables that are found in the base data set but not in the comparison data set.
LISTBASEOBS	lists all observations that are found in the base data set but not in the comparison data set.
LISTBASEVAR	lists all variables that are found in the base data set but not in the comparison data set.
LISTCOMP	lists all observations and variables that are found in the comparison data set but not in the base data set.
LISTCOMPOBS	lists all observations that are found in the comparison data set but not in the base data set.
LISTCOMPVAR	lists all variables that are found in the comparison data set but not in the base data set.
LISTEQUALVAR	prints a list of variables whose values are judged equal at all observations in addition to the default list of variables whose values are judged unequal.
LISTOBS	lists all observations that are found in only one data set.
LISTVAR	lists all variables that are found in only one data set.
METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>	specifies the method for judging the equality of numeric values.
OUT=*SAS-data-set*	names the output data set which contains differences between matching variables.
OUTALL	writes an observation for each observation in the BASE= and COMPARE= data sets.
OUTBASE	writes an observation to the output data set for each observation in the base data set, creating observations in which _TYPE_=BASE.
OUTCOMP	writes an observation to the output data set for each observation in the comparison data set, creating observations in which _TYPE_=COMP.
OUTDIF	writes an observation to the output data set for each pair of matching observations.
OUTNOEQUAL	suppresses the writing of an observation to the output data set when all values in the observation are judged equal.
OUTPERCENT	writes an observation to the output data set for each pair of matching observations.
OUTSTATS=*SAS-data-set*	writes summary statistics for all pairs of matching variables to the specified *SAS-data-set*.

Figure 6. PROC COMPARE options

OPTIONS WHICH RESTRICT THE OUTPUT

PROC COMPARE output can be quite lengthy especially if there are a large number of value differences found in specific variables. There are specific options for limiting and controlling the sections of output that are produced.

NOSUMMARY Option

The NOSUMMARY option suppresses the display of all summary reports and sections. This includes the dataset, variable, observation, and values comparison summary reports.

```
Proc Compare Base=SASHELP.PRDSAL2 Compare=SASHELP.PRDSAL3 NOSUMMARY;
Run;
```

With the summary sections of output suppressed, what appears in the SAS output are detail reports, such as the values comparison results section, and other details which normally appear in summary sections.

BRIEFSUMMARY Option

The BRIEFSUMMARY option produces a report with a condensed summary. However, it also suppresses the display of all default summary reports and sections. The detail sections, such as the values comparison results section will still appear in the output. In some situations, it may be worthwhile to use the BRIEFSUMMARY option, instead of the NOSUMMARY option.

```
Proc Compare Base=SASHELP.PRDSAL2 Compare=SASHELP.PRDSAL3 BRIEFSUMMARY;
Run;
```

NOVALUES Option

The NOVALUES option suppresses the values comparison results section. This option is quite handy because the values comparison results is a detailed record-by-record comparison of variable values found to be unequal. It can be very long, depending on the number of differences found.

```
Proc Compare Base=SASHELP.PRDSAL2 Compare = SASHELP.PRDSAL3 NOVALUES;
Run;
```

MAXPRINT Option

Instead of suppressing the entire section, you might choose to limit the number of differences to be printed. The MAXPRINT= option allows you to limit the number of differences to a reasonable level.

```
Proc Compare Base=PRDSAL2 Compare=PRDSAL3 MAXPRINT=25;
Run;
```

Either way, I recommend using the NOVALUES option to suppress the values comparison results section or the MAXPRINT option to limit the volume of SAS output produced.

OPTIONS TO SHOW DIFFERENCES IN THE BASE OR COMPARISON DATA SET

With PROC COMPARE output, you might want to include in the output only the variables, and observations found in one data set or the other. There are options which will output variables and observations found in the BASE data set, not the COMPARE data set. Conversely, there are options which will output the opposite.

OPTIONS WHICH RESTRICT OBSERVATIONS AND VARIABLES DISPLAYED

OPTION	OUTPUT
LISTBASE	VARIABLES AND OBSERVATIONS FOUND ONLY IN THE BASE DATA SET
LISTBASEVARS	VARIABLES FOUND IN THE BASE DATA SET ONLY
LISTBASEOBS	OBSERVATIONS FOUND IN THE BASE DATA SET ONLY
LISTCOMP	VARIABLES AND OBSERVATIONS FOUND ONLY IN THE COMPARE DATA SET
LISTCOMPVARS	VARIABLES IN THE COMPARE DATA SET ONLY
LISTCOMPOBS	OBSERVATIONS IN THE COMPARE DATA SET ONLY

Figure 7. PROC COMPARE Options which limit observations and variables

In Figure 7 above are the PROC COMPARE options which limit the output to specific elements of either the BASE or COMPARE data set.

To subset the output to include observations and variables in the BASE data set only, use the LISTBASE option. More specifically, you can focus on observations only using the LISTBASEOBS option, and variables only using the LISTBASEVARS option. These options include observations and variables specific to the BASE data set.

To include observations and variables in the COMPARE data set only use the LISTCOMP option.
You can choose observations unique to the COMPARE data set using the LISTCOMPOBS option, and variables unique to that same data set using the LISTCOMPVAR option.

APPLICATIONS OF PROC COMPARE

PARALLEL PROGRAMMING

Parallel programming is a type of project structure where a project is given to 2 programmers who work independently and write code from scratch to produce SAS output or a final data set from specifications. Also called double-independent programming, it's used often in clinical trials programming and in observational research as a validation method.

Since a particular project is normally given to a single programmer, by having 2 programmers work on the project, parallel programming provides another level of validation, which is one of the advantages of this structure. However, since the time and efforts of 2 programmers are utilized, one of the drawbacks of parallel programming is inefficiency and overuse of resources.

In parallel programming projects, PROC COMPARE plays a vital role in determining the status of the project outcomes and results. That is, whether the SAS output and data sets match or contain discrepancies.

```
104         Proc Compare Base=Prdsal2a Compare=Prdsal2b;
105         Run;

NOTE: There were 23040 observations read from the data set WORK.PRDSAL2A.
NOTE: There were 23040 observations read from the data set WORK.PRDSAL2B.
NOTE: PROCEDURE COMPARE used (Total process time):
      real time           0.10 seconds
      user cpu time       0.10 seconds
      system cpu time     0.01 seconds
      memory              5542.59k
      OS Memory           35760.00k
```

Figure 8. PROC COMPARE excerpt from the SAS Log.

In Figure 8 we have 2 versions of the PRDSAL SAS data set; PRDSAL2A, and PRDSAL2B. which hypothetically were produced by 2 programmers separately.

Consequently, PROC COMPARE was run to compare these 2 data sets, and the SAS code and log messages were copied from the SAS log. In most parallel programming projects, the data sets need to match in all aspects of the data set; variables, observations, data values, and variable attributes. Variable attributes include variable length, type, format, informat, and label.

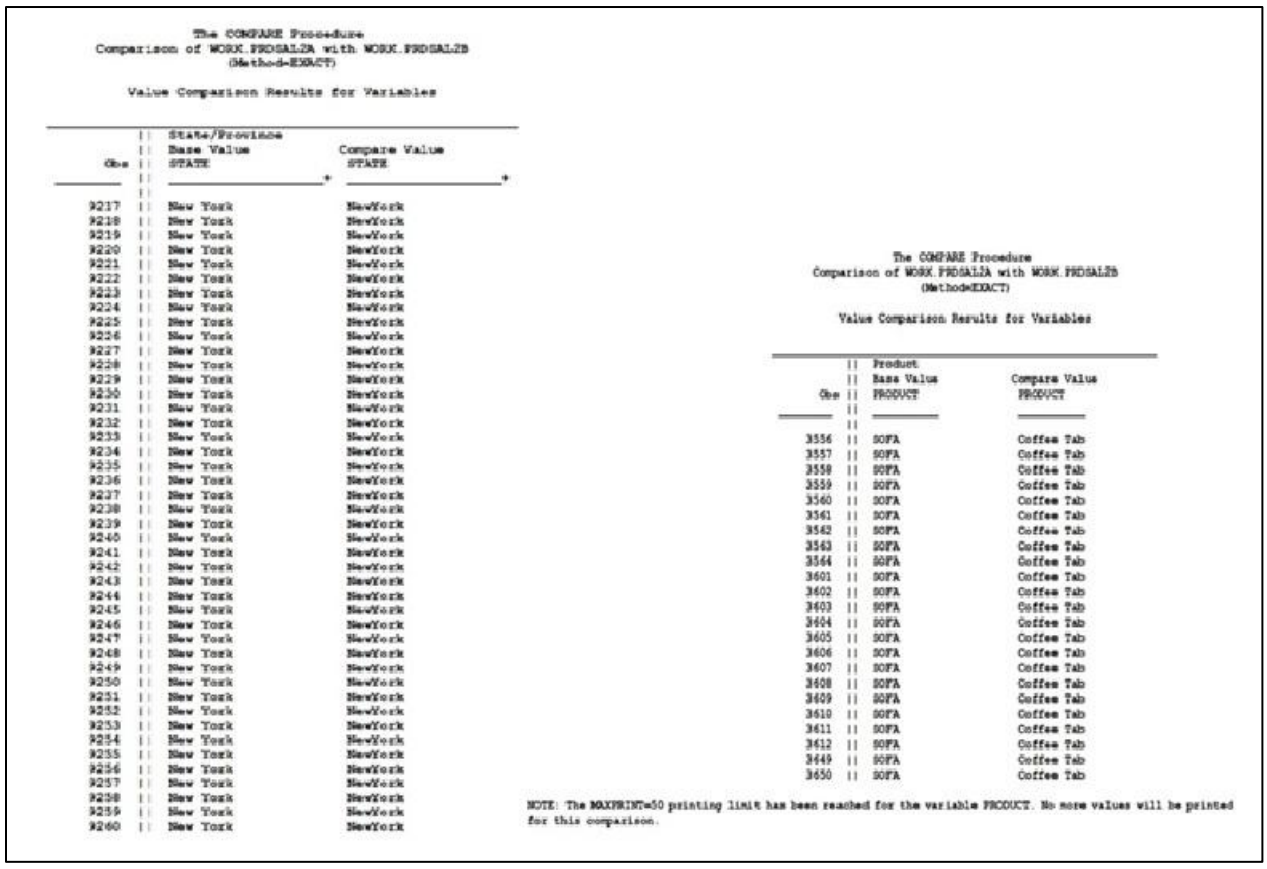
If there are any discrepancies found between the data sets, the SAS code of the programmers must be reviewed and debugged to locate the source of the discrepancy. PROC COMPARE is run again to assess status, and so on. This iterative process is performed over and over again until the data sets match.

In the PROC COMPARE output, specific types of discrepancies point to different sources of error, and different SAS processes.

For instance, discrepancies in number of observations align with data sets merged incorrectly in a DATA STEP Merge or PROC SQL JOIN. Consequently, the problem could stem from a sub-setting issue with the data in a WHERE or IF statement.

Variable discrepancies might mean a variable was omitted through a KEEP or DROP statement in one of the data sets. Value discrepancies indicate that a variable was derived or re-coded incorrectly, and the source is a bunch of IF-THEN statements. Perhaps the ELSE Keyword was left off an IF-THEN block.

In Figure 9 below, is the PROC COMPARE output from the SAS code provided in Figure 8.



The COMPARE Procedure
Comparison of WORK.PRDSALZA with WORK.PRDSALZB
(Method=EXACT)

Value Comparison Results for Variables

Obs	State/Province Base Value STATE	Compare Value STATE
9217	New York	NewYork
9218	New York	NewYork
9219	New York	NewYork
9220	New York	NewYork
9221	New York	NewYork
9222	New York	NewYork
9223	New York	NewYork
9224	New York	NewYork
9225	New York	NewYork
9226	New York	NewYork
9227	New York	NewYork
9228	New York	NewYork
9229	New York	NewYork
9230	New York	NewYork
9231	New York	NewYork
9232	New York	NewYork
9233	New York	NewYork
9234	New York	NewYork
9235	New York	NewYork
9236	New York	NewYork
9237	New York	NewYork
9238	New York	NewYork
9239	New York	NewYork
9240	New York	NewYork
9241	New York	NewYork
9242	New York	NewYork
9243	New York	NewYork
9244	New York	NewYork
9245	New York	NewYork
9246	New York	NewYork
9247	New York	NewYork
9248	New York	NewYork
9249	New York	NewYork
9250	New York	NewYork
9251	New York	NewYork
9252	New York	NewYork
9253	New York	NewYork
9254	New York	NewYork
9255	New York	NewYork
9256	New York	NewYork
9257	New York	NewYork
9258	New York	NewYork
9259	New York	NewYork
9260	New York	NewYork

The COMPARE Procedure
Comparison of WORK.PRDSALZA with WORK.PRDSALZB
(Method=EXACT)

Value Comparison Results for Variables

Obs	Product Base Value PRODUCT	Compare Value PRODUCT
3556	SOFA	Coffee Tab
3557	SOFA	Coffee Tab
3558	SOFA	Coffee Tab
3559	SOFA	Coffee Tab
3560	SOFA	Coffee Tab
3561	SOFA	Coffee Tab
3562	SOFA	Coffee Tab
3563	SOFA	Coffee Tab
3564	SOFA	Coffee Tab
3601	SOFA	Coffee Tab
3602	SOFA	Coffee Tab
3603	SOFA	Coffee Tab
3604	SOFA	Coffee Tab
3605	SOFA	Coffee Tab
3606	SOFA	Coffee Tab
3607	SOFA	Coffee Tab
3608	SOFA	Coffee Tab
3609	SOFA	Coffee Tab
3610	SOFA	Coffee Tab
3611	SOFA	Coffee Tab
3612	SOFA	Coffee Tab
3649	SOFA	Coffee Tab
3650	SOFA	Coffee Tab

NOTE: The MAXPRINT=50 printing limit has been reached for the variable PRODUCT. No more values will be printed for this comparison.

Figure 9. PROC COMPARE Output – Value Comparison Results.

The output in Figure 9 displays data value discrepancies in 2 variables; STATE and PRODUCT. The data values are character strings, and the differences are in how the character strings are coded. For STATE, there's a space embedded in the data value. For PRODUCT, data values have been coded incorrectly and truncated. Generally, this points to an issue with variable length.

In cases where there's a lot of discrepancies in PROC COMPARE output, it's a good practice to save the discrepancies in a separate SAS data set. You have the ability to do this using the OUT= option.

In Figure 10 below is the same PROC COMPARE example using the OUT= option. An output SAS data set PrdSal_Results is created using OUT= . In addition, several other options are used here; OUTNOEQUAL, OUTDIF and NOPRINT.

```
Proc Compare Base=Prdsal2a
             Compare=Prdsal2b
             Out=PrdSal_Results Outnoequal OutDif Noprint;
Run;
```

Figure 10. PROC COMPARE WITH OUT= option.

Creating an output SAS data set is a good practice when you have a lot of data value discrepancies, specifically, which are too numerous to include in standard SAS output.

MERGING SAS DATA SETS

Merging SAS data sets is a common data manipulation and programming task. Whether you use a DATA STEP Merge, or a PROC SQL Join, the BY variable used to perform the merge should have the same variable attributes. This programming situation presents a use case for PROC COMPARE.

Using PROC COMPARE, we can check and ascertain whether the BY variables align according to variable attributes of Variable Name, Length, Format, Type, and Label. Combining data sets using the DATA STEP Merge, the BY variables are required to have the same name.

In Figure 11 is a code example for comparing variable attributes of BY variables. To compare variable attributes, PROC COMPARE with VAR and WITH statements needs to be used.

```
Proc Compare Base=hlthcare.IP_2010_Claim  Compare=IP_2010_Claim
        Var BENE_ID;
        With BENE_ID;
Run;
```

Figure 11. PROC COMPARE example to compare attributes of BY variables.

If the BY variables don't have the same Type, Length, or Format, the results of the merge will be suspect and unreliable. For discrepancies in Variable Type or Format, SAS may print a warning message in the SAS log.

We're also interested in knowing what kind of relationship exists between the data sets. For example, if it's a one-to-one, one-to-many, or many-to-many. In other words, does either data set contain duplicates in BY variable values.

Using the BY variable in an ID statement, PROC COMPARE will inform us if duplicates exist.

Recall that the default behavior of PROC COMPARE is to compare observations by position or observation number. Specifying the BY variable in the ID statement overrides this default behavior and PROC COMPARE compares observations based on values of the BY variable.

In Figure 12 below is an example of PROC COMPARE with the ID statement from the SAS log.

```
81      Proc Sort Data = hlthcare.ip2010claim Out=ip2010claim;
82          By Bene_ID;
83      Run;

NOTE: There were 13916 observations read from the data set HLTHCARE.IP2010CLAIM.
NOTE: The data set WORK.IP2010CLAIM has 13916 observations and 36 variables.
NOTE: PROCEDURE SORT used (Total process time):
          real time       0.04 seconds
          user cpu time   0.05 seconds

85      Proc Sort Data = hlthcare.finder_attrib Out=finder_attrib;
86          By Bene_ID;
87      Run;

NOTE: There were 85872 observations read from the data set HLTHCARE.FINDER_ATTRIB.
NOTE: The data set WORK.FINDER_ATTRIB has 85872 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
          real time       0.04 seconds
          user cpu time   0.04 seconds

89      Proc Compare Base=ip2010claim Compare=finder_attrib LISTBASEOBS LISTCOMPOBS;
90          ID Bene_ID;
91          Var AT_NPI;
92          With PRFNPI;
93      Run;

WARNING: The data set WORK.IP2010CLAIM contains a duplicate observation at observation number 4.
NOTE: At observation 4 the current and previous ID values are:  BENE_ID=0007F12A492FD25D.
NOTE: Further warnings for duplicate observations in this data set will not be printed.
NOTE: There were 13916 observations read from the data set WORK.IP2010CLAIM.
NOTE: There were 85872 observations read from the data set WORK.FINDER_ATTRIB.

NOTE: PROCEDURE COMPARE used (Total process time):
          real time       0.77 seconds
          user cpu time   0.75 seconds
```

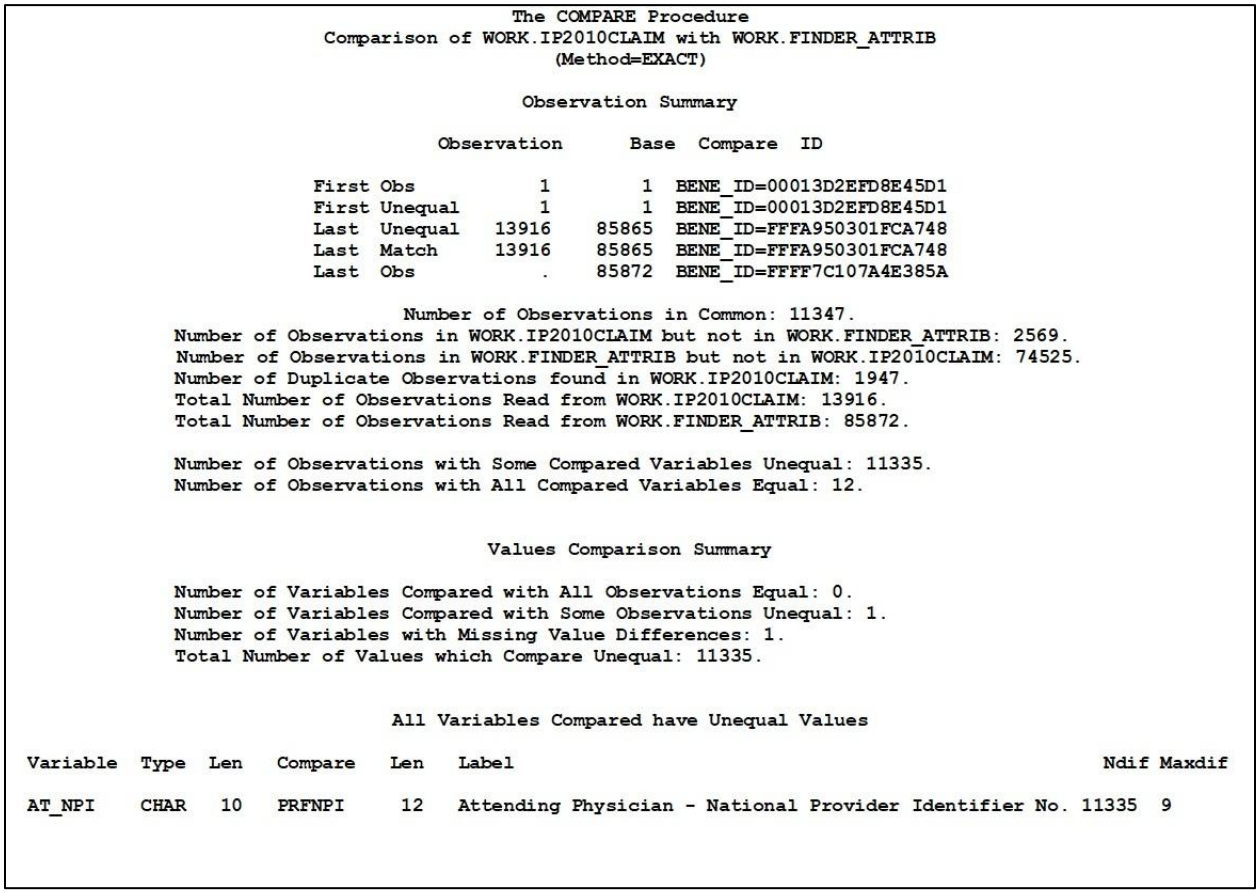
Figure 12. PROC COMPARE with ID statement – SAS Log Excerpt

To use them in an ID statement, the data sets need to be sorted on the BY variables prior to running PROC COMPARE. As shown in Figure 12, the data sets are sorted on the BY variable using PROC SORT.

Following this, PROC COMPARE is run on the data sets using the BY variable BENE_ID in the ID statement. The VAR and WITH statements are used in this example to limit the variables comparison to a single variable.

In Figure 12, a warning message is printed in the SAS log. The warning message indicates that one of the data sets contains a duplicate observation for observation number 4 for a specific value of BENE_ID, the BY variable. Through the warning about duplicates, SAS has notified the user that the type of merge is one-to-many.

The PROC COMPARE output from the code in Figure 12 is displayed in Figure 13.



```
                                The COMPARE Procedure
                        Comparison of WORK.IP2010CLAIM with WORK.FINDER_ATTRIB
                                        (Method=EXACT)

                                      Observation Summary

                                 Observation      Base  Compare  ID

                          First Obs           1         1  BENE_ID=00013D2EFD8E45D1
                          First Unequal       1         1  BENE_ID=00013D2EFD8E45D1
                          Last  Unequal   13916     85865  BENE_ID=FFFA950301FCA748
                          Last  Match     13916     85865  BENE_ID=FFFA950301FCA748
                          Last  Obs           .     85872  BENE_ID=FFFF7C107A4E385A

                                  Number of Observations in Common: 11347.
                Number of Observations in WORK.IP2010CLAIM but not in WORK.FINDER_ATTRIB: 2569.
                Number of Observations in WORK.FINDER_ATTRIB but not in WORK.IP2010CLAIM: 74525.
                Number of Duplicate Observations found in WORK.IP2010CLAIM: 1947.
                Total Number of Observations Read from WORK.IP2010CLAIM: 13916.
                Total Number of Observations Read from WORK.FINDER_ATTRIB: 85872.

                Number of Observations with Some Compared Variables Unequal: 11335.
                Number of Observations with All Compared Variables Equal: 12.


                                      Values Comparison Summary

                Number of Variables Compared with All Observations Equal: 0.
                Number of Variables Compared with Some Observations Unequal: 1.
                Number of Variables with Missing Value Differences: 1.
                Total Number of Values which Compare Unequal: 11335.


                                   All Variables Compared have Unequal Values

Variable  Type  Len   Compare   Len   Label                                                Ndif Maxdif

AT_NPI    CHAR   10   PRFNPI     12   Attending Physician - National Provider Identifier No. 11335  9
```

Figure 13. PROC COMPARE with ID Statement – Output

In Figure 13, there are several items of interest contained in the output.

The number of duplicate observations is printed in the output for one of the data sets. The output reports that the data set IP2010CLAIM had 1947 duplicates.

PROC COMPARE provides you with the number of observations the data sets had in common, as well as the number of observations found in the BASE data set but not the COMPARE data set, and vice versa.

In Figure 13, the output reports that the data sets had 11347 observations in common. But 2569 observations were in IP2010CLAIM, but not in FINDER_ATTRIB. Conversely, 74525 observations were in FINDER_ATTRIB, but not in IP2010CLAIM.

These numbers are the matches and non-matches from the merge, respectively. Normally, we'd have to execute the merge to produce the number of matches and non-matches. But using the ID statement, PROC COMPARE has provided this information for us beforehand.

CONCATENATING SAS DATA SETS

Another programming task which programmers often must complete is concatenating or appending SAS data sets. Concatenating data sets is combining data sets vertically, as opposed to merging which is combining them horizontally. This process is also called stacking data sets.

In BASE SAS, methods for concatenating include the DATA STEP, PROC APPEND, PROC SQL and PROC DATA SETS. Whatever method you choose there are requirements which the data sets have to meet.

Variables in the two data sets must have the same name. They should also have the same variable type, length and format. SAS may not execute the concatenation if the attributes don't align. If it does combine the data sets, the integrity and validity of the resulting data will be adversely impacted.

Concatenating data sets is another best practice case for PROC COMPARE. Programmers should check that the data sets and variables have consistent attributes prior to performing the concatenation.

In Figure 14 is an example of PROC COMPARE to check the variables of two SAS data sets, which need to be appended.

```
125     Proc Compare Base=US Compare=Mexico;
126     Run;

NOTE: There were 18432 observations read from the data set WORK.US.
NOTE: There were 4608 observations read from the data set WORK.MEXICO.
NOTE: PROCEDURE COMPARE used (Total process time):
   real time        0.15 seconds
   user cpu time    0.15 seconds
   system cpu time  0.01 seconds
   memory           4640.68k
   OS Memory        37696.00k
```

Figure 14. PROC COMPARE excerpts from SAS log.

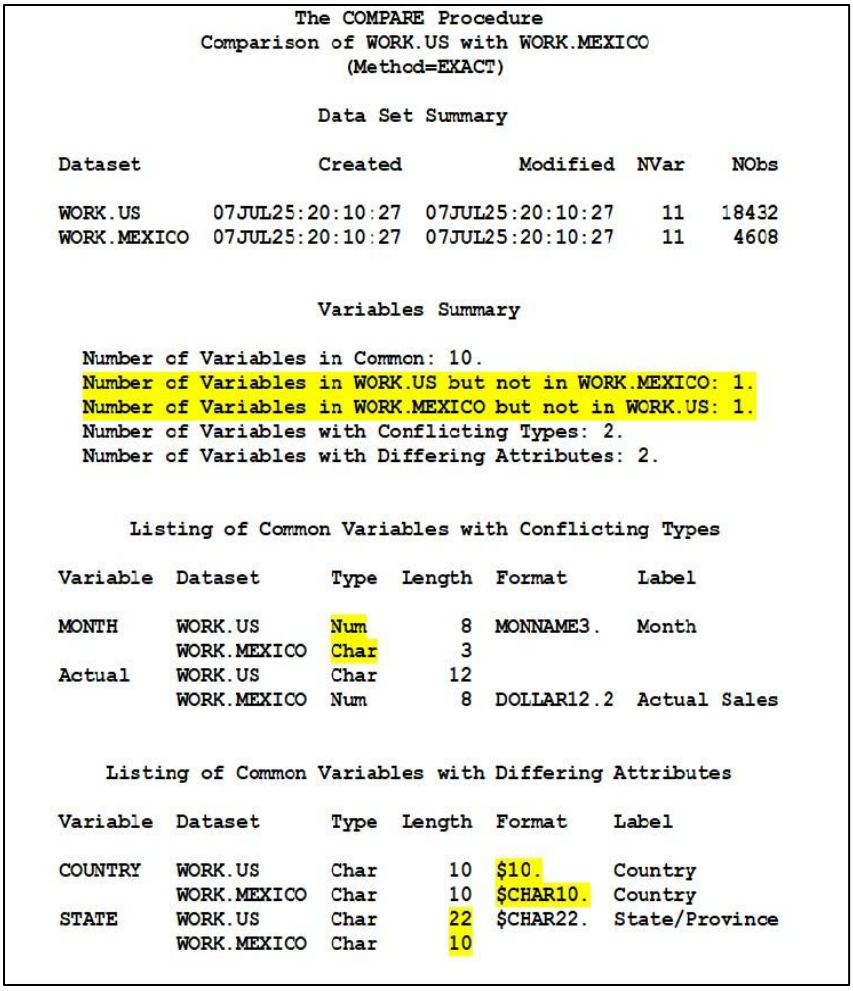
Figure 14 is an excerpt from the SAS log showing execution of the COMPARE procedure.

To discern variable attributes, you can also run PROC CONTENTS, which outputs a report with listings of variables contained in the data set and attributes for them.

The advantage of running PROC COMPARE is it performs an automated comparison of the data sets. With PROC CONTENTS, the programmer needs to perform the comparison manually.

In Figure 15 is the output from the code from Figure 14. Highlighted in the output are discrepancies which need to be resolved.

Missing variable discrepancies occur when a variable is present on one data set but not the other. In this case, each data set contains a single variable not found on the other data set.



The COMPARE Procedure
Comparison of WORK.US with WORK.MEXICO
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.US	07JUL25:20:10:27	07JUL25:20:10:27	11	18432
WORK.MEXICO	07JUL25:20:10:27	07JUL25:20:10:27	11	4608

Variables Summary

Number of Variables in Common: 10.
Number of Variables in WORK.US but not in WORK.MEXICO: 1.
Number of Variables in WORK.MEXICO but not in WORK.US: 1.
Number of Variables with Conflicting Types: 2.
Number of Variables with Differing Attributes: 2.

Listing of Common Variables with Conflicting Types

Variable	Dataset	Type	Length	Format	Label
MONTH	WORK.US	Num	8	MONNAME3.	Month
	WORK.MEXICO	Char	3		
Actual	WORK.US	Char	12		
	WORK.MEXICO	Num	8	DOLLAR12.2	Actual Sales

Listing of Common Variables with Differing Attributes

Variable	Dataset	Type	Length	Format	Label
COUNTRY	WORK.US	Char	10	$10.	Country
	WORK.MEXICO	Char	10	$CHAR10.	Country
STATE	WORK.US	Char	22	$CHAR22.	State/Province
	WORK.MEXICO	Char	10		

Figure 15. PROC COMPARE Output with variable attribute discrepancies

Several other types of discrepancies are present in Figure 15 output. Variable type and length discrepancies are the highest priority to resolve, because they lead to data validity issues.

The variable MONTH has a variable type discrepancy. It's numeric in one data set, but character in the other. Variable type discrepancies often involve SAS adopting the type from the first data set and generating missing values for the variable from the second data set.

The variable COUNTRY has a format discrepancy. However, since both variables are character, and have the same length, it's unlikely to involve an issue with the resulting data.

STATE has a variable length discrepancy. When length discrepancies occur, data values are often truncated.

Thus, the output has highlighted several issues which merit attention. These issues should be resolved through routine DATA STEP programming to ensure data quality.

RECONCILING VERSIONS OF SAS DATA SETS

Besides the ones we've discussed thus far, there are other programming projects and tasks where PROC COMPARE can play an effective role.

It's occasionally the case that programmers have to deal with multiple versions of SAS data sets which get created, and don't know the differences between them.

Also, there are SAS projects where you create and maintain a permanent SAS data set, called a master data set and update it periodically. The master data set gets updated with smaller SAS data sets, called transaction data sets.

The different versions of master data sets can be thought of as generation data sets. Creating generations data sets in SAS is another topic I only touch upon in passing in this paper.

In each of these cases, PROC COMPARE documents the differences between versions of SAS data sets, in variable data values, in observations, and so on.

The code in Figure 16 below displays the example of a mass shootings database which gets updated periodically, on an annual basis.

```
105         Proc Compare Base=Everytown_2023  Compare=Everytown_2025;
106         Run;

 NOTE: There were 284 observations read from the data set WORK.EVERYTOWN_2023.
 NOTE: There were 299 observations read from the data set WORK.EVERYTOWN_2025.
 NOTE: PROCEDURE COMPARE used (Total process time):
       real time           0.09 seconds
       user cpu time       0.09 seconds
       system cpu time     0.01 seconds
       memory              2063.40k
       OS Memory           32984.00k
```

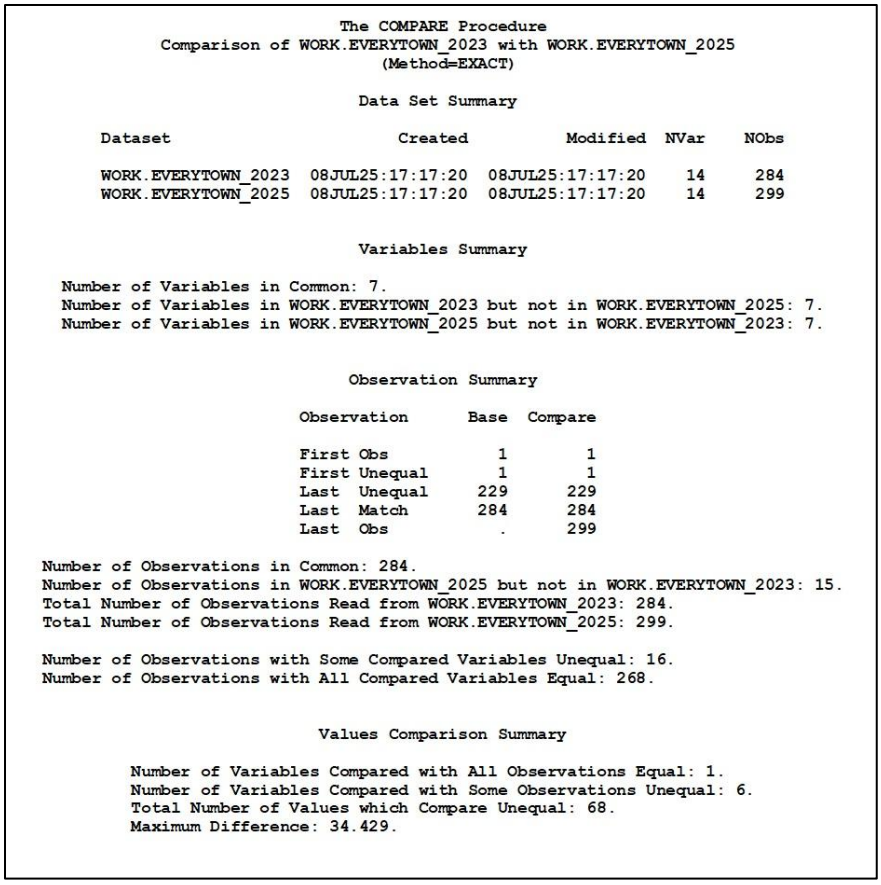
Figure 16. PROC COMPARE Example - excerpt from SAS log.

The database contains records of incidents of mass shootings occurring in the United States. When the database gets updated, new records are added to the database for new shooting occurrences.

In the example in Figure 16, PROC COMPARE is executed to compare two annual versions of the database. The version of the database from 2023 is compared with the updated version from 2025, respectively.

The PROC COMPARE output is provided in Figure 17 below.

The output shows differences in variables, observations, and variable values. The data sets have the same number of variables, although they contain different sets of them.



The COMPARE Procedure
Comparison of WORK.EVERYTOWN_2023 with WORK.EVERYTOWN_2025
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.EVERYTOWN_2023	08JUL25:17:17:20	08JUL25:17:17:20	14	284
WORK.EVERYTOWN_2025	08JUL25:17:17:20	08JUL25:17:17:20	14	299

Variables Summary

Number of Variables in Common: 7.
Number of Variables in WORK.EVERYTOWN_2023 but not in WORK.EVERYTOWN_2025: 7.
Number of Variables in WORK.EVERYTOWN_2025 but not in WORK.EVERYTOWN_2023: 7.

Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	229	229
Last Match	284	284
Last Obs	.	299

Number of Observations in Common: 284.
Number of Observations in WORK.EVERYTOWN_2025 but not in WORK.EVERYTOWN_2023: 15.
Total Number of Observations Read from WORK.EVERYTOWN_2023: 284.
Total Number of Observations Read from WORK.EVERYTOWN_2025: 299.

Number of Observations with Some Compared Variables Unequal: 16.
Number of Observations with All Compared Variables Equal: 268.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 6.
Total Number of Values which Compare Unequal: 68.
Maximum Difference: 34.429.

Figure 17. PROC COMPARE Output 1

The observation summary shows the 2025 updated version of the data set contains 15 additional observations over what the 2023 version of the data set contains.

The values comparison summary also shows differences in data values.

In reconciling SAS data sets, documenting differences in observations and data values are of primary importance. Although differences in variables are important, variable attribute differences are not. This is an example where the variable attribute section of the report can be suppressed. Although, PROC COMPARE may not have an option for doing that.

In Figure 18 is the variables with unequal values section of the output.

This section shows 6 variables which have differences in data values. Notice that 4 of the variables have data value differences in 12 of the observations. The remaining 2 variables have data value differences in 10 of the observations. To recall, the reported differences are only for matching observations.

The output also shows the maximum difference for Latitude and Longitude, the 2 numeric variables in the output. The remaining variables are character.

The COMPARE Procedure
Comparison of WORK.EVERYTOWN_2023 with WORK.EVERYTOWN_2025
(Method=EXACT)

Variables with Unequal Values

Variable	Type	Len	Label	Ndif	MaxDif
City	CHAR	24	City	12	
State	CHAR	2	State	10	
Latitude	NUM	8	Latitude	12	13.419
Longitude	NUM	8	Longitude	12	34.429
Narrative	CHAR	764	Narrative	12	
Last updated	CHAR	24	Last updated	10	

Figure 18. PROC COMPARE Output 2- Variables with unequal values

By examining the value comparison results section, we can surmise the source of the differences.

In Figure 19 is the values comparison results for variables section. It displays a record-by-record listing of observations which had value differences for the variable, CITY. We know that CITY had 12 records which had value differences.

Value Comparison Results for Variables

Obs	City Base Value	Compare Value
	City	City
90	Clarksburg	Hialeah
91	Hialeah	Clarksburg
106	Alturas	Indianapolis
107	Indianapolis	Alturas
146	Pike County	Appling
147	Appling	Pike County
148	Roswell	Orlando
149	Orlando	Roswell
176	Reading	Melcroft
177	Melcroft	Reading
228	Springfield	Moncure
229	Moncure	Springfield

Figure 19. Values comparison results for variables.

The variables contain the same data values, but not for the same observations. Thus, this is a case where the observations are in a different order.

From the output, we can discern that the source of the differences is that the data sets are sorted in a different order. This accounts for the differences in values rather than recoded variable values or some other source.

CONCLUSION

PROC COMPARE is an effective data validation construct for comparing two data sets within the BASE SAS package. Although the amount of output from PROC COMPARE can be of a large volume, the procedure features options which provide the ability to control and limit sections of output, and the amount of certain types of output produced. PROC COMPARE has applicability to a wide variety of programming projects, from parallel programming to merging data sets, to concatenating data sets and reconciling different versions of SAS data sets.

REFERENCES

Centers for Disease Control and Prevention, National Center for Health Statistics. "National Health and Nutrition Examination Survey, Questionnaires data sets and related documentation". Accessed 04/16/2020. URL-https://www.cdc.gov/nchs/nhanes/Search/ DataPage.aspx? Component= Dietary & CycleBeginYear =2009

Centers for Disease Control and Prevention, National Center for Health Statistics, National Health and Nutrition Examination Survey, Questionnaires data sets and related documentation, NHANES 2009-10 Dietary Data. "Dietary Supplements Use 30-Day – Individual Dietary Supplements (DSQIDS_F)". Accessed 04/16/2020. https://wwwn.cdc.gov/Nchs/Nhanes/2009-2010/DSQIDS_F.htm

Iyengar, Jayanth.. October 2011, "Can you decipher the code? If you can maybe you can break it". Proceedings of the Southeast SAS Users Group 2011 Conference, Alexandria, VA. Southeast SAS Users Group. Available at https://lexjansen.com/cgi-bin/xsl_transform.php?x=sesug2011 and https://analytics.ncsu.edu/ sesug/ 2011/ CC21.Iyengar.pdf

ACKNOWLEDGMENTS

The author would like to thank David Corliss, MWSUG 2025 Academic Chair, John LeBore, MWSUG 2025 Operations Chair; Richann Watson, Pharma and Life Sciences Section Chair, and the MWSUG 2025 Executive Committee and Conference team for accepting my abstract and paper and for organizing the conference.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:

Jay Iyengar
Data Systems Consultants LLC
datasyscon@gmail.com
https://www.linkedin.com/in/datasysconsult/

Jay Iyengar is director of Data Systems Consultants LLC. He is a SAS consultant, trainer and SAS Certified Advanced Programmer. He was co-leader and organizer of the Chicago SAS Users Group (WCSUG) from 2015-19. He's presented papers and training seminars at SAS Global Forum (SGF), and other regional and local SAS users' groups. His main industry experience is in Health care, Public Health and Pharmaceutical. He obtained his Bachelor's in Public Policy and Economics from Syracuse University and his master's degree from the American University.

TRADEMARK CITATION