DARE TO COMPARE
Tailoring PROC COMPARE Output

Maria Y. Reiss, Wyeth Pharmaceuticals, Collegeville, PA

INTRODUCTION

The COMPARE procedure is the SAS® tool for comparing two SAS data sets against each other. A tool to compare data sets is, of course, useful to SAS users. However, the default output from PROC COMPARE is very lengthy. To allow users to tailor their output, PROC COMPARE provides thirty-seven different options. This paper explains how to use the PROC COMPARE options to produce streamlined, focused reports of data set differences. This paper also explains two of the advanced features of PROC COMPARE:

- writing the output to a data set
- parsing the system return code from PROC COMPARE.

USING PROC COMPARE TO LOG AND CHECK DATA CHANGES

One application of PROC COMPARE is logging and checking changes to a SAS data set. You can use PROC COMPARE to compare the data set before and after editing to make sure that the desired changes were made correctly and that no unintentional changes were made. SAS 8.2 has a new audit trail feature designed for this purpose. However, the new audit trail has limitations. For example, the audit trail feature is "not appropriate for data sets that are rebuilt using the DATA step or the SQL procedure." [2] So you may prefer to use PROC COMPARE in place of the audit trail facility to log data changes.

WAYS THAT SAS DATA SETS CAN DIFFER

The PROC COMPARE documentation lists the following ways that SAS data sets can differ:

1. A variable can have different values in two observations being compared.
2. One data set can contain observations not in the other data set.
3. Data sets can contain different variables.
4. A variable can have structural differences in the two data sets such as different
   - informats
   - formats
   - lengths
   - labels
   - types
5. The labels of the two data sets can differ.
6. The types of the two data sets can differ.
7. The data sets can have BY group differences. For example, one data set can have a BY group that is not in the other data set. This type of difference will also appear in item #2 above (observation differences between data sets).

In this paper, I am not covering items #5, #6, and #7. Items #5 and #6 can be checked using PROC CONTENTS or PROC DATASETS, while item #7 is redundant with item #2.

DEFAULT PROC COMPARE

The simplest PROC COMPARE call you can make is:

```
proc compare base = base-data-set
             compare = compare-data-set;
run;
```

PROC COMPARE uses the terms "base data set" and "compare data set" to distinguish the two SAS data sets being compared.

This code will produce a lengthy report of all the differences between the two data sets. SAS will compare the data sets observation by observation in the order that the observations appear in the data sets. In other words, observation #1 from the base data set will be compared to observation #1 from the compare data set, etc.

The default PROC COMPARE call will produce the following summary reports:

- Data Set Summary
- Variables Summary
- Observation Summary
- Values Comparison Summary

After the four default summary reports, PROC COMPARE produces the reports:

- Variables with Unequal Values
- Value Comparison Results for Variables

CUSTOMIZING PROC COMPARE

The first step in customizing PROC COMPARE is to use the ID statement. The ID statement lists the key variables that uniquely identify the observations in your data sets. With the ID statement, PROC COMPARE compares observations that match on the ID variables.

Using the ID statement requires that the data sets be sorted by the ID variables or have an appropriate index. Also, if the ID variables do not uniquely identify the observations, PROC COMPARE will generate WARNING messages in the log for the duplicate observations. For the most accurate use of PROC COMPARE, choose ID variables that uniquely identify the observations.

```
proc compare base = base-data-set
             compare = compare-data-set;
   id key-variable-1 … key-variable-n;
run;
```

PROC COMPARE allows the use of a WHERE clause to subset the data being compared. For example:

```
proc compare base = base-data-set
             compare = compare-data-set;
   id key-variable-1 … key-variable-n;
   where variable-1 in ('value-1', 'value-2',
'value-3');
run;
```

You can also customize PROC COMPARE to limit the report output to

- a values comparison report.
- a report of observation differences.
- a report of variable mismatches.
- a report of structural differences between variables.

PRODUCING A VALUES COMPARISON REPORT

PROC COMPARE does not provide many options for shortening the report of value differences. The options that affect this type of report are:

briefsummary	Prints only a short comparison summary
nosummary	Suppresses the summary reports
transpose	Prints the value differences by observation, not by variable. Note that the transpose option does not shorten the output from PROC COMPARE. In fact, it may lengthen the output. However, I personally find the report easier to read in transpose format.
nomissbase	Judges a missing value in the base data set equal to any value.
nomisscomp	Judges a missing value in the comparison data set equal to any value.

To produce a values comparison report, use the code:

```
proc compare base = base-data-set
             compare = compare-data-set
             nosummary transpose;
   id id-variables;
run;
```

An example of the output is in **Appendix 1**.

For each value difference, PROC COMPARE prints a row listing the ID variables and the variable's value in both the BASE and COMPARE data sets. For character variables, PROC COMPARE only prints the first 20 characters of the variable's value. If the difference in the variable's value in the BASE and COMPARE data sets lies past the 20th character, the variable will still be listed in the Values Comparison report, but the value difference will not be displayed. The only way that PROC COMPARE will display character value differences past the 20th character is by writing the PROC COMPARE output out to a data set using the OUT= option.

PRODUCING A REPORT OF OBSERVATION DIFFERENCES

PROC COMPARE provides the following options to allow you to produce a report of observation differences.

nosummary	Suppresses the summary reports
novalues	Suppresses the value comparison results.
listobs	Lists all observations found in only one data set.
listall	Lists all variables and observations found in only one data set.
listbase	Lists all variables and observations found only in the base data set.
listbaseobs	Lists all observations found only in the base data set.
listcomp	Lists all variables and observations found only in the comparison data set.
listcompobs	Lists all observations found only in the comparison data set.

To produce a report of observation differences, use the code:

```
proc compare base = base-data-set
             compare = compare-data-set
             nosummary novalues listobs;
   id id-variables;
run;
```

An example of the output is in **Appendix 2**.

PRODUCING A REPORT OF VARIABLE MISMATCHES IN THE DATA SETS

PROC COMPARE provides the following options to that allow you to produce a report of variable mismatches.

nosummary	Suppresses the summary reports
novalues	Suppresses the value comparison results.
listvar	Lists all variables found in only one data set.
listall	Lists all variables and observations found in only one data set.
listbase	Lists all variables and observations found only in the base data set.
listbasevar	Lists all variables found only in the base data set.
listcomp	Lists all variables and observations found only in the comparison data set.
listcompvar	Lists all variables found only in the comparison data set.

To produce a report of variable mismatches, use the code:

```
proc compare base = base-data-set
             compare = compare-data-set
             nosummary novalues listvar;
   id id-variables;
run;
```

An example of the output is in **Appendix 3**.

If there are variable mismatches, then the code will generate a report of the variable mismatches and a report of the structural differences between variables. However, if there are no variable mismatches, the code will not generate a report of the structural differences between variables even if such differences exist.

PRODUCING A REPORT OF STRUCTURAL DIFFERENCES BETWEEN VARIABLES

There are two ways to generate of report of structural differences (such as informat, format, length, label, and type) between variables.

1. Run a PROC COMPARE with the novalues option. There is no way to limit the output to **only** list structural differences between variables.

2. Process PROC CONTENTS output.[3]

```
proc contents data = base-data-set
     out = base-contents noprint; run;
proc contents data = compare-data-set
     out = compare-contents noprint; run;
proc compare base = base-contents
             compare = compare-contents
             briefsummary transpose;
   id name ;
   var length type format informat label;
run;
```

Note: In this code, the variable names are case-sensitive. If the variable name is lower-case in the base data set and upper-case in the compare data set, then they will not match up with the "id name" statement. They will be considered as two separate variables.
An example of the output is in **Appendix 4**.

OTHER USEFUL PROC COMPARE OPTIONS

PROC COMPARE has two options, VAR and WITH, that can be used together to

1. Compare variables with different names in two data sets.

```
proc compare base = base-data-set
             compare = compare-data-set
             nosummary;
  var base-variable;
  with compare-variable;
  title 'Comparison of Vars with Different Names';
run;
```

2. Compare a single variable with multiple variables.

```
proc compare base = base-data-set
             compare = compare-data-set
             nosummary;
  var base-variable base-variable;
  with compare-variable-#1 compare-variable-#2;
  title1 'Compare One Variable (base-variable)';
  title2 'with Two Variables (compare-variable-
#1 and compare-variable-#2)';
run;
```

3. Compare a variable with another variable in the same data set.

```
proc compare base = base-data-set
             nosummary;
    var base-variable-#1;
    with base-variable-#2;
    title 'Comparison of Variables in the Same
Data Set';
run;
```

FUZZ
You can enter values from 0 through 1 for the FUZZ option. In the Values Comparison Report, PROC COMPARE will print 0 for any variable value that is less than the FUZZ value. It will also print a blank for the difference column in the Values Comparison Report. However, it will still print an entry in the Values Comparison Report for every variable value and difference that is less than the FUZZ number. So this option will not shorten the Values Comparison Report.

See **Appendix 5** for an example of a Values Comparison Report with FUZZ=0.5.

CRITERION
This option "specifies the criterion for judging the equality of numeric values." (from SAS OnlineDOC®) The Online SAS document states that the default value of criterion is 0.00001.
I have found this option useful when comparing SAS data sets that have been created on different operating systems. The numeric variables in data sets created on different operation systems may differ slightly in value due to differences in numeric precision on the different operation systems. As stated in SAS Technical Support document 654, "It is not uncommon to get slightly different results between operating systems whose floating point representation components differ (i.e. MVS and PC, MVS and UNIX). Some problems with numeric precision arise because the underlying instructions that each operating system uses to do addition, multiplication, division, etc. are slightly different. There is no standard method for doing computations since all operating systems attempt to compute numbers as accurately as possible." [4]

If you are using PROC COMPARE to compare numeric variables from data sets created on different operation systems, and your PROC COMPARE report is very lengthy due to miniscule precision differences in the numeric variables, you can use the CRITERION option to shorten the output. The PROC COMPARE documentation recommends setting CRITERION to -1000 to suppress listings of differences that are the results of differences in the machine level of precision.

ADVANCED PROC COMPARE FEATURES

Writing to an Output Data Set

PROC COMPARE provides options that allow you to write the output to an output data set. The output data set can contain four different types of observations, each type distinguished by the value of the _TYPE_ variable. The four types are:
1. BASE – the observation in the base data set.
2. COMPARE – the observation in the compare data set.
3. DIF – an observation showing the differences between the base and compare data sets. For character variables, the DIF observation contains a period or an X for every character in the variable. A period indicates that the variables match at that position in the variable. An X indicates that the variables do not match at that position in the variable. For numeric variables, the DIF observation contains an E if the variables match. If the numeric variables do not match, the DIF observation contains the difference in the values in the base and compare data sets.
4. PERCENT – an observation showing the percentage differences in values between the base and compare data sets.

As stated earlier, the Values Comparison Report from PROC COMPARE does not display character value differences past the 20th character. The only way to display character value differences past the 20th character is by writing the PROC COMPARE output out to a data set. [5]

Example:

Dataset #1 contains the following data:
```
ID  NAME     ADDRESS
011 REISS    REALLY REALLY REALLY REALLY LONG ADDRESS
026 WILLIAMS 272 Monmouth Drive
028 SMITH    Unknown
```

Dataset #2 contains the following data:
```
ID  NAME       ADDRESS
011 REISS      REALLY REALLY REALLY REALLY LONG NAME
026 WILLIAMS   272 Monmouth Drive
028 SMITH      Unknown
```

To produce a report showing data differences past the 20th character of a character variable, use the code:

```
proc compare base = data-set-#1
             compare = data-set-#2
             noprint out=comp_out
             outbase outcomp outdiff;
   id id;
   title1 "Comparison of DS1 and DS2";
run;

proc print data = comp_out width=min;
  where id = 11;
run;
```

The output from this PROC PRINT (for ID 011 showing ADDRESS variable only) is:

```
_TYPE_   ADDRESS
BASE     REALLY REALLY REALLY REALLY LONG ADDRESS
COMPARE  REALLY REALLY REALLY REALLY LONG NAME
DIF      ................................XXXXXXX...
```

The BASE observation shows the data from the base data set (data set #1). The COMPARE observation shows the data from the compare data set (data set #2). The DIF observation shows the differences between the base and compare data sets. Periods in the character variables indicate that the variables match at that position in the variable. Xs in the character variables indicate that the variables do not match at that position in the variable. [5]

Return Code from PROC COMPARE

PROC COMPARE returns a return code in the system macro variable SYSINFO. The values in SYSINFO are stored so that the binary value of SYSINFO indicates which differences exist between the two data sets. You can determine the data set differences by testing SYSINFO using bit-testing within a DATA step.

The values of SYSINFO are:

Condition	Value of SYSINFO	Bit	Value to Use for Bit-Test
Data set labels differ	$1 = 2^0$	0	‘1’b
Data set types differ	$2 = 2^1$	1	‘1.’b
Variable has different infomat	$4 = 2^2$	2	‘1..’b
Variable has different format.	$8 = 2^3$	3	‘1…’b
Variable has different length.	$16 = 2^4$	4	‘1….’b
Variable has different label.	$32 = 2^5$	5	‘1…..’b
Base data set has observation not in comparison.	$64 = 2^6$	6	‘1……’b
Comparison data set has observation not in base.	$128 = 2^7$	7	‘1…….’b
Base data set has BY group not in comparison.	$256 = 2^8$	8	‘1……..’b
Comparison data set has BY group not in base.	$512 = 2^9$	9	‘1………’b
Base data set has variable not in comparison.	$1024 = 2^{10}$	10	‘1……….’b
Comparison data set has variable not in base.	$2048 = 2^{11}$	11	‘1………..’b
A value comparison was unequal.	$4096 = 2^{12}$	12	‘1…………’b
Conflicting variable types.	$8192 = 2^{13}$	13	‘1………….’b
BY variables do not match.	$16384 = 2^{14}$	14	‘1…………..’b
Fatal error: comparison not done.	$32768 = 2^{15}$	15	‘1……………’b

An example of using bit-testing of the SYSINFO value from PROC COMPARE is:

```
proc compare … etc.

%let rc=&sysinfo;
data _null_;
  * Check if SYSINFO has a 1 in the 2^12
    position of its binary value.;
  if &rc=‘1…………’b then
   put ‘At least 1 value comparison was unequal’;
run;
```

EXAMPLES

I. USING PROC COMPARE TO LOG DATA CHANGES

You can use PROC COMPARE to create a log of data edits.

Example:

Version 01 of the PAYROLL data contains the following data:

```
ID      SALARY
011     245
026     269
028     374
034     333
057     582
060     100
```

The data manager creates Version 02 of the PAYROLL data with edits:

```
data payrollv02;
  set payrollv01;

  if id=057 then salary=600;
  if id=026 then delete;
run;
```

To produce a report of the edits, use the code:

```
title1 "Differences Between PAYROLLV02 and
PAYROLLV01";
proc compare base = payrollv01
             compare = payrollv02
             briefsummary listall transpose;
  id id;
run;
```

See **Appendix 6** for the output from this PROC COMPARE.

II. USING PROC COMPARE TO FIND UNINTENDED CHANGES

When merging two data sets that contain a common variable with different lengths in the two data sets, SAS will take the length from the first data set in the MERGE statement. If the variable has a shorter length in the first data set, this can cause values from the second data set to be truncated in the merged data set.

Example:

Data Set 1- LNAME Has Length $10

```
ID    LNAME        SALARY
11    REISS          376
26    WILLIAMS       900
28    SMITH          777
```

Data Set 2 - LNAME Has Length $17

```
ID    LNAME
11    REISS
26    WILLIAMS
28    SMITH
34    HOSKINSON-ALVAREZ
57    JONES
60    BROWN
```

These two data sets are merged using the code:

```
data mergedata;
  merge ds1
        ds2;
  by id;
run;
```

The variable LNAME in the resulting data set MERGEDATA has length $10. The PROC COMPARE code

```
proc compare base = ds2
             compare = mergedata
             briefsummary;
   id id;
   title1 "Comparison of DS2 and MERGEDATA";
run;
```

produces output that shows that the variable LNAME has been truncated for the record with ID 34.

```
NOTE: Values of the following 1 variables
compare unequal: LNAME

Value Comparison Results for Variables

________________________________________________
         ||  Base Value           Compare Value
     ID  ||  LNAME                 LNAME
_______  ||  _________________     __________
         ||
     34  ||  HOSKINSON-ALVAREZ     HOSKINSON-
________________________________________________
```

III. USING PROC COMPARE TO DETERMINE CHANGES THAT WOULD BE MADE WITH UPDATE STATEMENT

You can use the PROC COMPARE option NOMISSCOMP to print a list of the changes that will be made to a master data set in a DATA step UPDATE statement.

For example, the DATA step statements

```
data newpay;
  update payroll increase;
  by id;
run;
```

will update the master data set PAYROLL with values from the transaction data set INCREASE according to the rules of the UPDATE statement. [6]

To see in advance what changes will be made to the master data set PAYROLL, run the following PROC COMPARE prior to running the DATA step UPDATE statement.

```
proc compare base = payroll
             compare = increase
             nosummary nomisscomp;
  id id;
run;
```

An example of the output is in **Appendix 7**.

TAILORED PROC COMPARE OUTPUT VERSUS FULL PROC COMPARE OUTPUT

If you want to produce a full report of the differences between two data sets, it is more efficient to run the default PROC COMPARE. However, if you are interested in narrowing the output of your report to a Values Comparison report, an Observation Differences report, or a Variable Differences report, then it is more efficient to use PROC COMPARE options to limit your report.

In a test comparing two data sets where each data set contained 26,192 observations and 40 variables on a Solaris 5.8 operating system, the default PROC COMPARE took 2.36 seconds of real CPU time. Comparing the same data sets using PROC COMPARE options to specifically produce a Values Comparison report, an Observation Differences report, and a Variable Differences report took 3.17 seconds of real CPU time.

See **Appendix 8** for a listing of a macro that produces tailored PROC COMPARE output. This macro also uses bit-testing of the return code from PROC COMPARE to produce a summary report of data set differences.

CONCLUSION

The COMPARE procedure provides many options for tailoring its output. This paper provides an explanation of the most useful options. This paper also provides examples of the practical usage of the COMPARE procedure.

REFERENCES

(1) SAS Institute Inc. (1999), *SAS OnlineDOC®, Version 8,* Chapter 9, The COMPARE Procedure; pp 221-266, Cary, NC: SAS Institute Inc.

(2) Thielbar, Melinda. “Hey, Who Changed My Data?” *SAS Bits & Bytes.* February 2003. <http://support.sas.com/sassamples/bitsandbytes/0208_audit.htm> (May 6, 2003).

(3) Crawford, Peter. “Comparing Variables of Two Data Sets and Outputting the Differences” The FAQChest January 2003 <http://www.faqchest.com/appl/sas-l/sas-03/sas-0301/sas-030110/sas03011016_26153.html >

(4) “Numeric Precision 101” *SAS Institute Inc. Technical Support Documents* <http://ftp.sas.com/techsup/download/technote/ts654.html>

(5) “How can I use PROC COMPARE to produce a report that shows the differences between two character values past the 20th character?” *SAS Institute Inc. Technical Support Documents* <http://support.sas.com/techsup/technote/ts440.pdf>

(6) SAS Institute Inc. (1999), *SAS OnlineDOC®, Version 8,* Chapter 24, Reading, Combining, and Modifying SAS Data Sets; pp 348-357, Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Maria Y. Reiss
Wyeth Research
500 Arcola Road
Collegeville, PA 19426
Email: reissm@wyeth.com
Phone: (484) 865-5694

APPENDIX 1 – Sample Values Comparison Report

```
                 The COMPARE Procedure
        Comparison of WORK.PHYSEXAM1 with WORK.PHYSEXAM2
                     (Method=EXACT)

              Comparison Results for Observations

DRUGID=0468 PROTID=220 PATIENT=000101 VISITDT=20SEP2000:00:00:00
 bdysys=:
Variable    Base Value       Compare         Diff.        % Diff
  RDCMID     559885501     560679801        794300      0.141868
 DCMQGID       2697201       2676901        -20300     -0.752632
 RCDCMID     559885501     560679801        794300      0.141868
   DCNUM    2096785401    2096874801
     DCM  PHYSICAL EXA    DEMOGRAPHY
 QLVALUE   Screening,2     Screening
 DCMDESC  PHYSICAL EXA    DEMOGRAPHY
    var1        Test 1        Test 2
```

APPENDIX 2 – Sample Observation Differences Report

```
                 The COMPARE Procedure
        Comparison of WORK.PHYSEXAM1 with WORK.PHYSEXAM2
                     (Method=EXACT)

              Comparison Results for Observations

Observation 193 in WORK.PHYSEXAM1 not found in WORK.PHYSEXAM2:
DRUGID=0468 PROTID=220 PATIENT=000102 VISITDT=05OCT2000:00:00:00
 bdysys=.

Observation 194 in WORK.PHYSEXAM1 not found in WORK.PHYSEXAM2:
DRUGID=0468 PROTID=220 PATIENT=000102 VISITDT=05OCT2000:00:00:00
 bdysys=Abdomen.

Observation 195 in WORK.PHYSEXAM1 not found in WORK.PHYSEXAM2:
DRUGID=0468 PROTID=220 PATIENT=000102 VISITDT=05OCT2000:00:00:00
 bdysys=Chest.

Observation 196 in WORK.PHYSEXAM1 not found in WORK.PHYSEXAM2:
DRUGID=0468 PROTID=220 PATIENT=000102 VISITDT=05OCT2000:00:00:00
 bdysys=Ears.

NOTE: Data set WORK.PHYSEXAM1 contains 4 observations not in WORK.PHYSEXAM2.
NOTE: Values of the following 14 variables compare unequal: DCNUM DCM RDCMID DCMQGID QLVALUE RPTSN
REGIMEN CPENM VISIT SUBVST INVEST DCMDESC RCDCMID var1
```

APPENDIX 3 – Sample Variable Mismatches Report

```
         Variables Found in Only One of the Data Sets physexam1 and physexam2

                          The COMPARE Procedure
                 Comparison of WORK.PHYSEXAM1 with WORK.PHYSEXAM2
                              (Method=EXACT)

           Listing of Variables in WORK.PHYSEXAM1 but not in WORK.PHYSEXAM2

                 Variable  Type  Length  Format  Informat  Label

                 STINT     Char      30  $30.    $30.      planned study interval name
                 TGTPNT    Num        8                    interval midpoint


           Listing of Variables in WORK.PHYSEXAM2 but not in WORK.PHYSEXAM1

     Variable  Type  Length  Format   Informat     Label

     UNIQSUBJ  Char      21                        GPSLH - Unique Subject/Patient (the latest)
```

```
UNIQSITE  Char      10                        GPSLH - Unique Study Site (the latest)
AGE       Num        8                        Age
AGEU      Char      20  $20.     $20.         Age Unit


          Listing of Common Variables with Differing Attributes

                       Variable  Dataset         Type  Length

                       var1      WORK.PHYSEXAM1  Char      12
                                 WORK.PHYSEXAM2  Char       8

NOTE: Data set WORK.PHYSEXAM1 contains 32 observations not in WORK.PHYSEXAM2.
NOTE: Values of the following 14 variables compare unequal: DCNUM DCM RDCMID DCMQGID QLVALUE RPTSN
REGIMEN CPENM VISIT SUBVST INVEST DCMDESC RCDCMID var1
```

APPENDIX 4 – Sample Report of Structural Differences Between Variables

```
              Comparing PROC CONTENTS of Datasets physexam1 and physexam2
          To Find Variable Differences in LENGTH, TYPE, FORMAT, INFORMAT, and LABEL

                                 The COMPARE Procedure
                  Comparison of WORK._BASE_CONTENTS with WORK._COMP_CONTENTS
                                    (Method=EXACT)

                           Comparison Results for Observations

                     NAME=var1:
                     Variable    Base Value       Compare         Diff.        % Diff
                       LENGTH     12.000000      8.000000     -4.000000    -33.333333

NOTE: Data set WORK._BASE_CONTENTS contains 2 observations not in WORK._COMP_CONTENTS.
NOTE: Data set WORK._COMP_CONTENTS contains 30 observations not in WORK._BASE_CONTENTS.
NOTE: Values of the following 1 variables compare unequal: LENGTH
```

APPENDIX 5 – Sample Report Using the FUZZ Option

Note: In this example, the variable VAR1 has a value of 0.1 in the Base data set and a value of 0.4 in the Compare data set. FUZZ is set to 0.5

```
                                 The COMPARE Procedure
                       Comparison of WORK.PHYSEXAM1 with WORK.PHYSEXAM2
                                    (Method=EXACT)

                           Comparison Results for Observations

                     DRUGID=0468 PROTID=220 PATIENT=000101 VISITDT=20SEP2000:00:00:00
                      bdysys=:
                     Variable    Base Value       Compare         Diff.        % Diff
                         var1             0             0                  300.000000

NOTE: Values of the following 1 variables compare unequal: var1
```

APPENDIX 6 – Using PROC COMPARE to Log Data Changes

```
                          Differences Between PAYROLLV02 and PAYROLLV01     17:03 Wednesday, April 23, 2003   1

                                 The COMPARE Procedure
                     Comparison of WORK.PAYROLLV01 with WORK.PAYROLLV02
                                    (Method=EXACT)

                           Comparison Results for Observations
          Observation 2 in WORK.PAYROLLV01 not found in WORK.PAYROLLV02:
          ID=26.

          ID=57:
          Variable    Base Value      Compare        Diff.      % Diff
            SALARY    582.000000   600.000000    18.000000    3.092784

NOTE: Data set WORK.PAYROLLV01 contains 1 observations not in WORK.PAYROLLV02.
NOTE: Values of the following 1 variables compare unequal: SALARY
```

APPENDIX 7 – Using PROC COMPARE To Determine Changes with Update Statement

```
                                The COMPARE Procedure
                         Comparison of WORK.PAYROLL with WORK.INCREASE
                                    (Method=EXACT)

NOTE: Values of the following 1 variables compare unequal: SALARY

                              Value Comparison Results for Variables

                             __________________________________________________________
                                       ||       Base    Compare
                                   ID  ||     SALARY     SALARY      Diff.     % Diff
                              _______  ||  _________  _________  _________  _________
                                       ||
                                   11  ||   245.0000   376.0000   131.0000    53.4694
                                   60  ||   100.0000   900.0000   800.0000   800.0000
                             __________________________________________________________
```

APPENDIX 8 – Macro to Produce Tailored PROC COMPARE Output

```
%macro compare (ds1 = , ds2 = , idvars = );
  %************************************************************************;
  %* Get difference in number of observations in Base and Compare data sets.  *;
  %************************************************************************;
  %getmacro (numobs);

  %local num_obs_diff;
  %let num_obs_diff=%eval(%sysfunc(abs(%eval(%numobs(&ds1)) - %eval(%numobs(&ds2)))));


  %************************************;
  %* Local Macro Variable Constants.  *;
  %************************************;
  %local val_diff;            %* Flag indicating if a value difference exists between ds1 and ds2. ;
  %let val_diff = FALSE;
  %local obs_diff;            %* Flag indicating if an observation difference exists between ds1 & ds2.;
  %let obs_diff = FALSE;

%* Create local macro vars for the 16 possible return code values for PROC COMPARE return code;
  %do i = 1 %to 16;
      %local compare_sysinfo_&i;        %* Bit flag ;
      %local compare_condition_&i;      %* Corresponding text description for the bit flag. ;
  %end;

  %let compare_sysinfo_1  = '1'b;
  %let compare_condition_1 = "Data Set Labels Differ.";
  %let compare_sysinfo_2  = '1.'b;
  %let compare_condition_2 = "Data Set Types Differ.";
  %let compare_sysinfo_3  = '1..'b;
  %let compare_condition_3 = "Variable Has Different Informat.";
  %let compare_sysinfo_4  = '1...'b;
  %let compare_condition_4 = "Variable Has Different Format.";
  %let compare_sysinfo_5  = '1....'b;
  %let compare_condition_5 = "Variable Has Different Length.";
  %let compare_sysinfo_6  = '1.....'b;
  %let compare_condition_6 = "Variable Has Different Label.";
  %let compare_sysinfo_7  = '1......'b;
  %if %eval(&num_obs_diff)=1 %then
      %let compare_condition_7 = "%upcase(&ds1) Data Set Has &num_obs_diff Obs. not in %upcase(&ds2) Data Set.";
  %else %let compare_condition_7 = "%upcase(&ds1) Data Set Has &num_obs_diff Obs. Not in %upcase(&ds2) Data Set.";

  %let compare_sysinfo_8  = '1.......'b;
  %if %eval(&num_obs_diff)=1 %then
  %let compare_condition_8 = "%upcase(&ds2) Data Set Has &num_obs_diff Observation Not in %upcase(&ds1) Data Set.";
  %else %let compare_condition_8 = "%upcase(&ds2) Data Set Has &num_obs_diff Observations Not in %upcase(&ds1) Data Set.";

  %let compare_sysinfo_9  = '1........'b;
  %let compare_condition_9 = "%upcase(&ds1) Data Set has BY Group Not in %upcase(&ds2) Data Set.";
  %let compare_sysinfo_10 = '1.........'b;
```

```
%let compare_condition_10 = "%upcase(&ds2) Data Set Has BY Group Not in %upcase(&ds1) Data Set.";
%let compare_sysinfo_11 = '1..........'b;
%let compare_condition_11 = "%upcase(&ds1) Data Set Has At Least One Variable Not in %upcase(&ds2) Data Set.";
%let compare_sysinfo_12 = '1...........'b;
%let compare_condition_12 = "%upcase(&ds2) Data Set Has At Least One Variable Not in %upcase(&ds1) Data Set.";
%let compare_sysinfo_13 = '1............'b;
%let compare_condition_13 = "At Least One Value Comparison Was Unequal.";
%let compare_sysinfo_14 = '1.............'b;
%let compare_condition_14 = "Conflicting Variable Types.";
%let compare_sysinfo_15 = '1..............'b;
%let compare_condition_15 = "BY Variables Do Not Match.";
%let compare_sysinfo_16 = '1...............'b;
%let compare_condition_16 = "Fatal Error: Comparison Not Done.";

%*********************************;
%* Print report of differences.  *;
%*********************************;
%include 'var_compare.sas';
%include 'obs_compare.sas';
%include 'val_compare.sas';

%*** Get next title number.;
proc sql noprint;
   select max(number) into :titlenum from sashelp.vtitle;

%*** Create VARIABLE DIFFERENCES REPORT. ;
title%eval(&titlenum+1) "VARIABLE DIFFERENCES REPORT";
title%eval(&titlenum+2) "NOTE: Base = &ds1, Compare = &ds2, ID Variables = &idvars";
%var_compare (ds1=&ds1, ds2=&ds2, idvars=&idvars)

%*** Save system return code from PROC COMPARE. ;
%let rc=&sysinfo;
%put ************ RC= &rc;

%*** Create summary report of all differences in LOG.;
data _null_;
   length numdiff 8;
   numdiff = 0;

   page;
   put;
   put '-------------------------------------------------------------------------------------------';
   put " Summary of Differences Between %upcase(&ds1) and %upcase(&ds2) Data Sets ";
   put '-------------------------------------------------------------------------------------------';

   if &rc = 0 then do;
      put " There Are No Differences - %upcase(&ds1) and %upcase(&ds2) Are Identical.";
   end;

   %if &rc > 0 %then %do;
       %do i = 1 %to 16;
          if &rc = &&compare_sysinfo_&i then do;
             numdiff+1;
             put ' ' numdiff +(-1) ') ' &&compare_condition_&i;
          end;
       %end;
   %end;

   put '-------------------------------------------------------------------------------------------';
   put;

   if &rc = &compare_sysinfo_13 then do;
        call symput ("val_diff",'TRUE');
   end;

   if &rc = &compare_sysinfo_7 or &rc = &compare_sysinfo_8 then do;
      call symput ("obs_diff",'TRUE');
   end;
run;

%if &obs_diff = TRUE %then %do;
    title%eval(&titlenum+1) "OBSERVATION DIFFERENCES REPORT";
    title%eval(&titlenum+2) "NOTE: Base = &ds1, Compare = &ds2, ID Variables = &idvars";
    %obs_compare (ds1=&ds1, ds2=&ds2, idvars=&idvars)
```

```
  %end;

  %if &val_diff = TRUE %then %do;
      title%eval(&titlenum+1) "VALUES COMPARISON REPORT";
      title%eval(&titlenum+2) "NOTE: Base = &ds1, Compare = &ds2, ID Variables = &idvars";
      %val_compare (ds1=&ds1, ds2=&ds2, idvars=&idvars)
  %end;

%mend compare;


%*-----------------------------------------------------*;
%* Macro to compare VARIABLES in two datasets.         *;
%*-----------------------------------------------------*;
%macro var_compare (ds1=          /* Name of Base dataset */
                   ,ds2=          /* Name of Compare dataset */
                   ,idvars=       /* List ID variables for datasets - must make all obs. unique*/ );

   %*** Get next title number.;
  proc sql noprint;
     select max(number) into :titlenum from sashelp.vtitle;

  title%eval(&titlenum+1) "Variables Found in Only One of the Datasets &ds1 and &ds2";
  proc compare base=&ds1
               compare=&ds2 nosummary novalues listvar;
    id &idvars;
  run;

  proc contents data=&ds1 out=_base_contents noprint; run;
  proc contents data=&ds2 out=_comp_contents noprint; run;

  title%eval(&titlenum+1) "Comparing PROC CONTENTS of Datasets &ds1 and &ds2";
  title%eval(&titlenum+2) "To Find Variable Differences in LENGTH, TYPE, FORMAT, INFORMAT, and LABEL";
  proc compare base=_base_contents
               compare=_comp_contents
               briefsummary transpose;
    id name;
    var length type format informat label;
  run;
%mend var_compare;


%*-----------------------------------------------------*;
%* Macro to compare OBSERVATIONS in two datasets.      *;
%*-----------------------------------------------------*;
%macro obs_compare (ds1=          /* Name of Base dataset */
                   ,ds2=          /* Name of Compare dataset */
                   ,idvars=       /* List ID variables for datasets - must make all obs. unique*/);

  %*** Print out observations in only one data set.;
  proc compare base=&ds1
               compare=&ds2 nosummary novalues listobs;
    id &idvars;
  run;


%mend obs_compare;


%*-----------------------------------------------------*;
%* Macro to compare variable values in two datasets.   *;
%*-----------------------------------------------------*;
%macro val_compare (ds1=          /* Name of Base dataset */
                   ,ds2=          /* Name of Compare dataset */
                   ,idvars=       /* List ID variables for datasets - must make all obs. unique*/ );

  %*** Compare variable values in two datasets.;
  proc compare base=&ds1
               compare=&ds2
               nosummary transpose ;
    id &idvars;
  run;

%mend val_compare;
```