PharmaSUG 2021 - Paper QT-147

Overcoming the Challenge of SAS Numeric Date Comparisons

Shikha Sreshtha, Covance by Labcorp

ABSTRACT

The outputs from the PROC COMPARE are tedious to work with especially when handling SAS numeric dates in huge datasets. The SAS dates are essentially numbers, that is, the number of days between 01JAN1960 to the input date and with the FORMAT statement, the date variables are only displayed in one of the standard date formats. When comparing numeric dates in clinical formats of standard e8601da10. using PROC COMPARE, all we see are frustrating little stars in the list file. To the rescue, are these huge array of options that the COMPARE procedure comes with. When faced with record-SAS date differences, most programmers turn to temporary data sub-setting and patches of code to determine the key variables to locate where the issue lies. All this effort to identify the one tiny needle in stack of hay and inform the 'base' programmer of the intended variable value. With the help of a handy SAS user-defined macro, we can do a side-by-side comparison of the SAS numeric dates in the output dataset and shift the 'compare' programmers focus to do a comprehensive review. The output dataset from the macro is concise and displays the actual numeric dates that makes the interpretation easy. The macro is designed to highlight the key variables for which the record SAS date differences exist and do a side-by-side comparison in a dataset format rather than a list file format.

INTRODUCTION

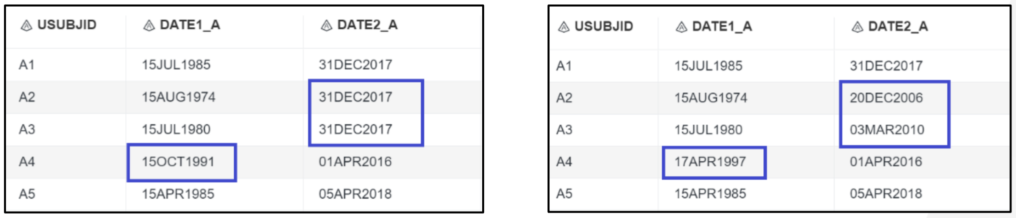
To confirm the authenticity of the the ‘base’ programmers’ dataset, the validation programmer braces themselves to jump in the compare field and look for the discrepancies. If the validation process isn’t already a hassle as is, the comparison of SAS numeric dates is in itself a notorious task. Upon further checks, it’s clear that the SAS numeric date comparison is not cumbersome for all formats but definitely for dates in **e8601da10.** formats. When dealing with SAS dates, users should be very well aware of the difference between how a SAS date is stored and how it is displayed. SAS date are stored as a value that represents the number of days between January 1, 1960, and the input date. Now this difference value will have to be formatted to display the dates in a desired manner.

LET THE COMPARISON BATTLE BEGIN

PROC COMPARE is a powerful tool to perform validation checks. No matter how mammoth or tiny your dataset be, it isn’t stamped a hundred percent correct unless the dataset undergoes the validation process. With a standard compare the programmer’s task is to check for the list file to detect the differences in the ‘base’ and ‘validation’ datasets and pass on the information to the base programmer. Before this step, the validator also ascertains WHY the mismatch occured to assure WHAT changes will result in the correct variable value. This task is a pain when SAS dates have to be compared; the issue that we’ll look in deep in the next sections. In all the further references the dates are in e8601da10. format where we encountered the issue. The COMPARE procedure works very well with dates in other formats like DATE9., DDMMYY10. and more.

CHARACTER DATES COMPARISON

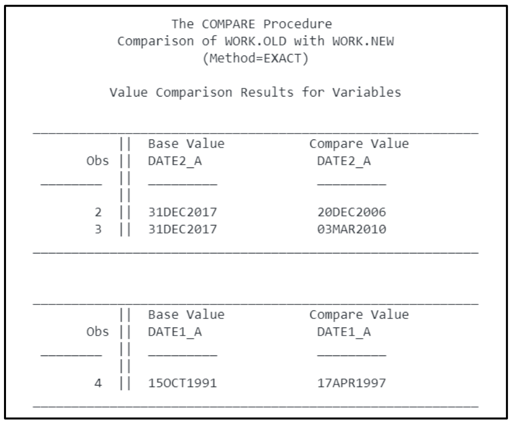
Since we are concentrating only on the date handling in the SAS datasets, here is a comparison of raw character date variables. Below list file comparison is satisfying to look at since one can visually have a look at the observation/row number where the issue lies and the two different date values in the variable DATE1_A and DATE2_A.



USUBJID	DATE1_A	DATE2_A
A1	15JUL1985	31DEC2017
A2	15AUG1974	31DEC2017
A3	15JUL1980	31DEC2017
A4	15OCT1991	01APR2016
A5	15APR1985	05APR2018

USUBJID	DATE1_A	DATE2_A
A1	15JUL1985	31DEC2017
A2	15AUG1974	20DEC2006
A3	15JUL1980	03MAR2010
A4	17APR1997	01APR2016
A5	15APR1985	05APR2018

Figure 1. Base and compare datasets with character date values

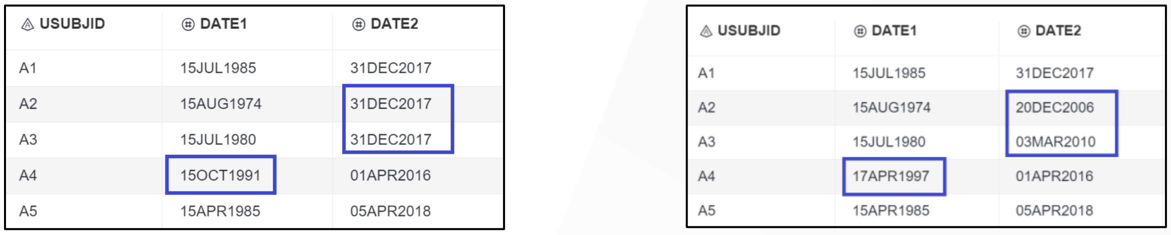


```
                       The COMPARE Procedure
                 Comparison of WORK.OLD with WORK.NEW
                           (Method=EXACT)

                Value Comparison Results for Variables

   ____________________________________________________________
             ||  Base Value           Compare Value
         Obs ||  DATE2_A               DATE2_A
    ________ ||  _________             _________
             ||
           2 ||  31DEC2017             20DEC2006
           3 ||  31DEC2017             03MAR2010
   ____________________________________________________________


   ____________________________________________________________
             ||  Base Value           Compare Value
         Obs ||  DATE1_A               DATE1_A
    ________ ||  _________             _________
             ||
           4 ||  15OCT1991             17APR1997
   ____________________________________________________________
```

Figure 2. List file generated with compare procedure

NUMERIC DATES COMPARISON IN A FORMAT OTHER THAN E8601DA10. FORMAT

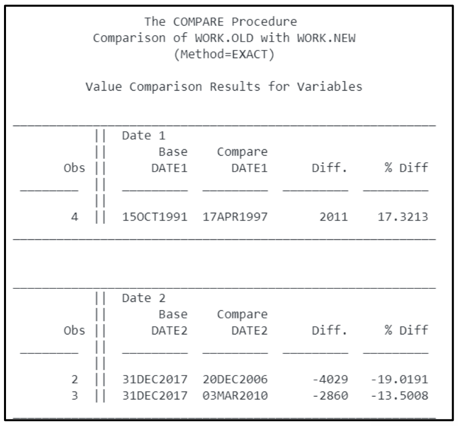
This too is a passable example since the SAS dates in numeric DATE9. format is compared without any warnings and a clean appeasing list file is generated.



USUBJID	DATE1	DATE2
A1	15JUL1985	31DEC2017
A2	15AUG1974	31DEC2017
A3	15JUL1980	31DEC2017
A4	15OCT1991	01APR2016
A5	15APR1985	05APR2018

USUBJID	DATE1	DATE2
A1	15JUL1985	31DEC2017
A2	15AUG1974	20DEC2006
A3	15JUL1980	03MAR2010
A4	17APR1997	01APR2016
A5	15APR1985	05APR2018

Figure 3. Base and compare datasets with numeric date values

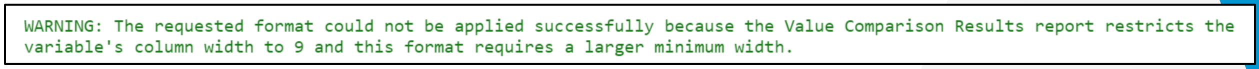


```
                   The COMPARE Procedure
             Comparison of WORK.OLD with WORK.NEW
                       (Method=EXACT)

            Value Comparison Results for Variables

 __________________________________________________________
          ||  Date 1
          ||       Base    Compare
      Obs ||      DATE1      DATE1      Diff.     % Diff
 ________ ||  _________  _________  _________  _________
          ||
        4 ||  15OCT1991  17APR1997       2011    17.3213
 __________________________________________________________


 __________________________________________________________
          ||  Date 2
          ||       Base    Compare
      Obs ||      DATE2      DATE2      Diff.     % Diff
 ________ ||  _________  _________  _________  _________
          ||
        2 ||  31DEC2017  20DEC2006      -4029   -19.0191
        3 ||  31DEC2017  03MAR2010      -2860   -13.5008
 __________________________________________________________
```

Figure 4. List file generated with compare procedure

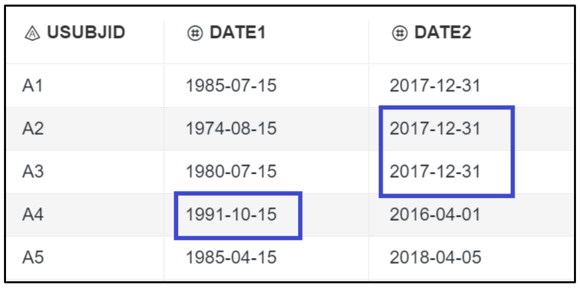
NUMERIC DATES COMPARISON IN E8601DA10. FORMAT

For two datasets differing in SAS numeric dates in e860da10. format (blue boxes), an awful looking list file comes up along with a warning. At least the warning gives us a fair idea of WHY the appearance of stars in the list file.

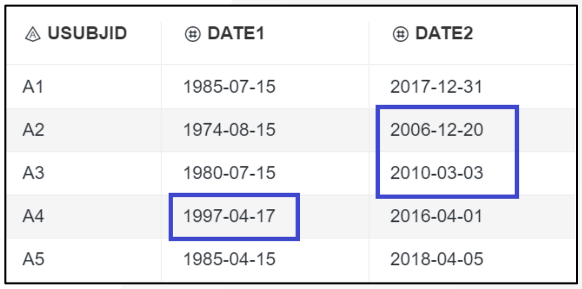


```
WARNING: The requested format could not be applied successfully because the Value Comparison Results report restricts the
variable's column width to 9 and this format requires a larger minimum width.
```

Figure 5. Warning in log file when comparing numeric dates in e8601da10. Format



USUBJID	DATE1	DATE2
A1	1985-07-15	2017-12-31
A2	1974-08-15	2017-12-31
A3	1980-07-15	2017-12-31
A4	1991-10-15	2016-04-01
A5	1985-04-15	2018-04-05

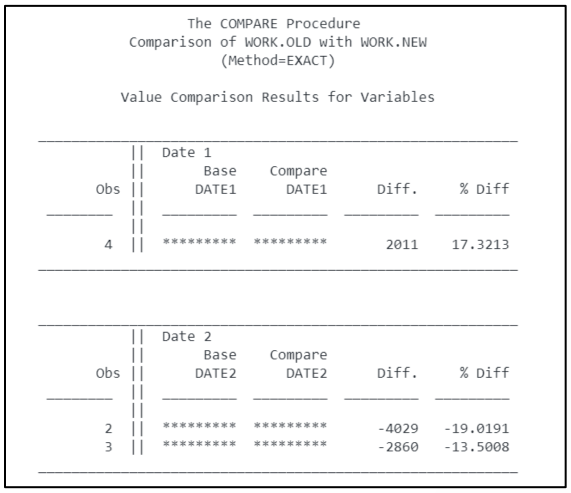


USUBJID	DATE1	DATE2
A1	1985-07-15	2017-12-31
A2	1974-08-15	2006-12-20
A3	1980-07-15	2010-03-03
A4	1997-04-17	2016-04-01
A5	1985-04-15	2018-04-05

Figure 6. Base and compare datasets with numeric date values

SAS numeric dates, that are formatted in e8601da10. for display gives us the below comparison in list file. Let’s see what information can we gather from this comparison sheet;

- there is mismatch at observation no. 4 for DATE1 variable. The difference between the mismatched dates being 2011 days but **actual dates not visible**
- mismatches at observation 2 and 3 for DATE2 variable. The difference between the mismatched dates being -4020 and -2860 days respectively (negative difference implies the comparison dates are smaller than base dates) and **actual dates not visible**



```
                    The COMPARE Procedure
           Comparison of WORK.OLD with WORK.NEW
                      (Method=EXACT)

            Value Comparison Results for Variables

__________________________________________________________
            ||  Date 1
            ||       Base    Compare
        Obs ||      DATE1      DATE1      Diff.     % Diff
 ________   ||  _________  _________  _________  _________
            ||
          4 ||  *********  *********       2011    17.3213
__________________________________________________________


__________________________________________________________
            ||  Date 2
            ||       Base    Compare
        Obs ||      DATE2      DATE2      Diff.     % Diff
 ________   ||  _________  _________  _________  _________
            ||
          2 ||  *********  *********      -4029   -19.0191
          3 ||  *********  *********      -2860   -13.5008
__________________________________________________________
```

Figure 7. List file generated with compare procedure

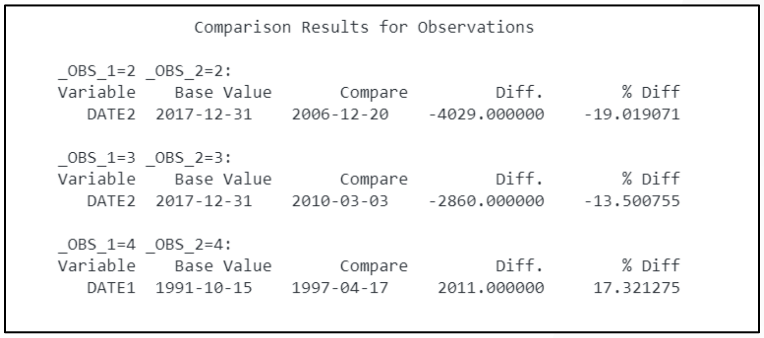
The comparison file does not give us enough information to take next steps. As a validator it would be ideal to know related information like the observation number and details of other variables on the mismatched row. Since there is minimal communication between the base and validation programmer throughout the buildup of final datasets it becomes imperative to give just the right amount of information to the base programmer to make corrections to the mismatched date difference.

Here the user has two options:

- Give the mismatched date variables another format and compare. This is the most efficient way to work around the issue, if you have the flexibility to temporary format the dates in the 'base' programmer dataset, bearing in mind the format should be such which does not yield a variable column length restriction while comparison.

- Another alternative would be to make use of the COMPARE procedure options which gives us a better look of the comparison file. In the below snapshots we have used few options which are detailed
    - OUTBASE option writes an observation to the output data set for each observation in the base data set
    - OUTCOMP option writes an observation to the output data set for each observation in the comparison data set
    - OUTDIF option writes an observation to the output data set for each pair of matching observations. The values in the observation include values for the differences between the values in the pair of observations.
    - TRANSPOSE option prints the reports of value differences by observation instead of by variable.

```
proc compare base=old compare=new listall outbase outcomp outdif transpose
             out=comp_out;
run;
```



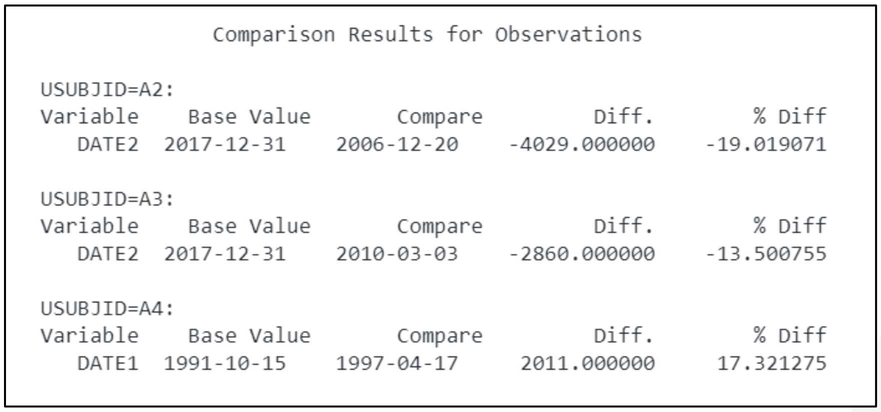
```
              Comparison Results for Observations

 _OBS_1=2 _OBS_2=2:
 Variable    Base Value       Compare          Diff.        % Diff
    DATE2  2017-12-31    2006-12-20    -4029.000000    -19.019071

 _OBS_1=3 _OBS_2=3:
 Variable    Base Value       Compare          Diff.        % Diff
    DATE2  2017-12-31    2010-03-03    -2860.000000    -13.500755

 _OBS_1=4 _OBS_2=4:
 Variable    Base Value       Compare          Diff.        % Diff
    DATE1  1991-10-15    1997-04-17     2011.000000     17.321275
```

Figure 8. List file generated with various options in compare procedure

It is nice to finally have a look at the list file with actual dates for the mismatched rows.

We can pitch in another option of the ID statement which displays the key variable for easy detection.

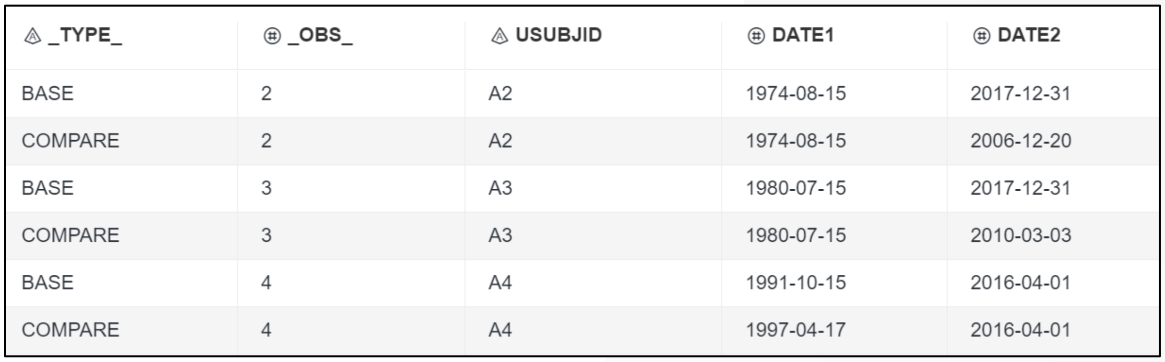
```
proc compare base=old compare=new listall outbase outcomp outdif transpose
             out=comp_out;
             id usubjid;
run;
```



```
                Comparison Results for Observations

USUBJID=A2:
Variable    Base Value       Compare          Diff.        % Diff
   DATE2  2017-12-31    2006-12-20    -4029.000000    -19.019071

USUBJID=A3:
Variable    Base Value       Compare          Diff.        % Diff
   DATE2  2017-12-31    2010-03-03    -2860.000000    -13.500755

USUBJID=A4:
Variable    Base Value       Compare          Diff.        % Diff
   DATE1  1991-10-15    1997-04-17     2011.000000     17.321275
```

Figure 9. Enhanced list file generated with ID option in compare procedure

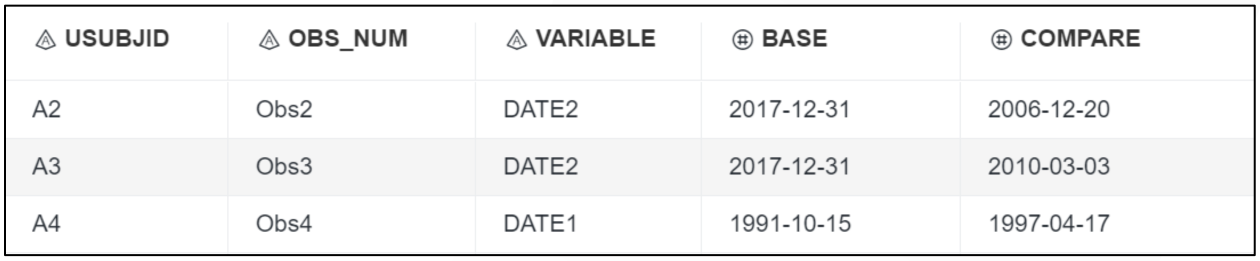
The only drawback being that the TRANSPOSE option prints the differences in the list file where we can see the side by side comparison of mismatched dates in list file. With the **OUT=** and **OUTNOEQUAL**= option one gets to see the mismatched dates in output dataset but in a vertical fashion as below. If OUTNOEQUAL option is omitted, then the output dataset lists out all available data in the base and comparison dataset which might not be necessary.



TYPE	_OBS_	USUBJID	DATE1	DATE2
BASE	2	A2	1974-08-15	2017-12-31
COMPARE	2	A2	1974-08-15	2006-12-20
BASE	3	A3	1980-07-15	2017-12-31
COMPARE	3	A3	1980-07-15	2010-03-03
BASE	4	A4	1991-10-15	2016-04-01
COMPARE	4	A4	1997-04-17	2016-04-01

Figure 10. All date values for variable DATE1/DATE2 are displayed.

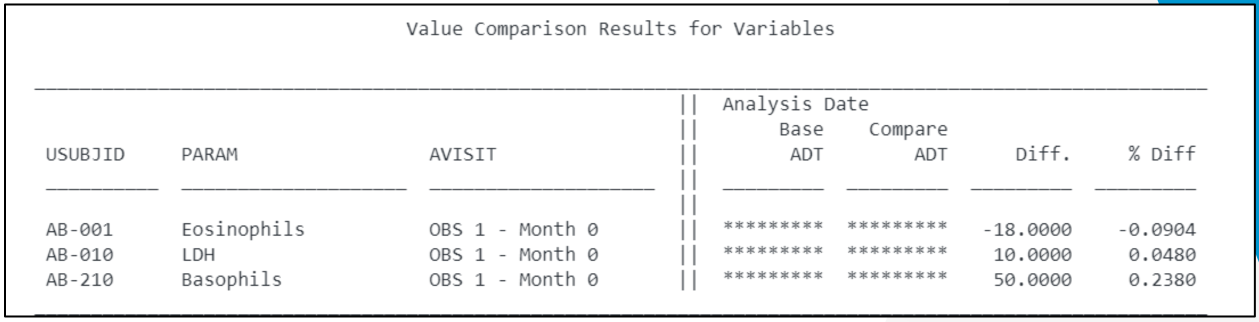
Under a time-crunch scenario this would likely be excess information that you might not be interested in. A small statement in macro gets rid of this unwelcome information and the final dataset will look as below.



USUBJID	OBS_NUM	VARIABLE	BASE	COMPARE
A2	Obs2	DATE2	2017-12-31	2006-12-20
A3	Obs3	DATE2	2017-12-31	2010-03-03
A4	Obs4	DATE1	1991-10-15	1997-04-17

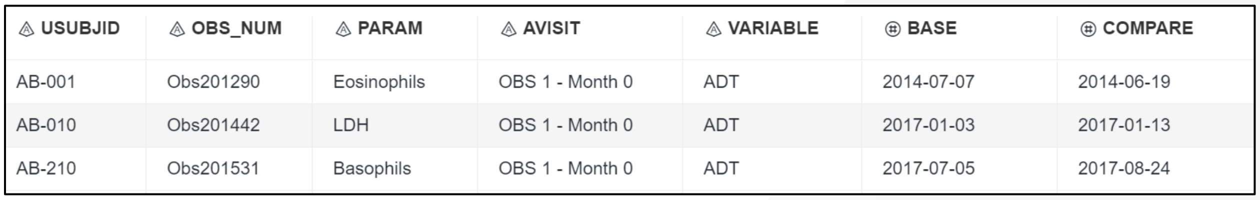
Figure 11. Only mismatched date values for DATE1/DATE2 variable are displayed

This macro usage can be extended to multiple identifiers and multiple numeric date variables. Below is an example of the same macro used on a huge Laboratory dataset.



```
                                  Value Comparison Results for Variables

 _____________________________________________________________________________________________________
                                                          ||  Analysis Date
                                                          ||       Base    Compare
 USUBJID     PARAM                 AVISIT                 ||        ADT        ADT      Diff.     % Diff
 __________  ____________________  ____________________   ||  _________  _________  _________  _________
                                                          ||
 AB-001      Eosinophils           OBS 1 - Month 0        ||  *********  *********   -18.0000    -0.0904
 AB-010      LDH                   OBS 1 - Month 0        ||  *********  *********    10.0000     0.0480
 AB-210      Basophils             OBS 1 - Month 0        ||  *********  *********    50.0000     0.2380
 _____________________________________________________________________________________________________
```

Figure 12. Output from PROC COMPARE with 3 variables (USUBJID, PARAM, AVISIT) in the ID statement.



USUBJID	OBS_NUM	PARAM	AVISIT	VARIABLE	BASE	COMPARE
AB-001	Obs201290	Eosinophils	OBS 1 - Month 0	ADT	2014-07-07	2014-06-19
AB-010	Obs201442	LDH	OBS 1 - Month 0	ADT	2017-01-03	2017-01-13
AB-210	Obs201531	Basophils	OBS 1 - Month 0	ADT	2017-07-05	2017-08-24

Figure 13. Output from the macro

CONCLUSION

To deliver a high quality output with correct information, the validator is burdened with the task of doing a comprehensive review of the datasets. Over time we have noted that there are very many variables dependent on dates. So, if the dates are correctly mapped, the dependent variables will be populated accordingly. A tiny macro can actually help you in this regard and help focus the validators attention towards other in-depth checks of the dataset.

APPENDIX

Source code for %COMP_DT set-up:

```
%macro comp_dt(basdsn=, compdsn=, key=, key1=, vars=);
    proc compare base=&basdsn compare=&compdsn listall outbase outcomp
                       out=comp_out(keep=_type_ _obs_ &key &key1 &vars);
          var &key &key1 &vars;
    run;

    proc sort data=comp_out out=comp_outz;
        by &key _obs_ &key1;
    run;

    proc transpose data=comp_outz out=comp_tran;
          by &key _obs_ &key1;
          id _type_;
          var &vars;
    run;

    data comp_tran_dt(drop=_label_ _obs_);
          retain &key &key1 obs_num _name_ base compare;
          set comp_tran;
          length obs_num $200.;
          obs_num='Obs'||strip(put(_obs_,8.));
          if base=. and compare=. then delete;
          if base ne compare;
          rename _name_= variable;
          format &vars e8601da10.; ** optional format statement **;
    run;

%mend comp_dt;
```

Macro requirement:
basdsn – input base programmer dataset
compdsn – input comparison programmer dataset
key – input primary key variable
key1 – input list of secondary key variables
vars – list of SAS numeric dates to be compared

CONTACT INFORMATION

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

Shikha Sreshtha
shikha.sreshtha@covance.com