MWSUG 2018 - Paper SP-069

What's the Difference? Using the PROC COMPARE to find out.

Larry Riggen, Indiana University, Indianapolis, IN

ABSTRACT

We are often asked to determine what has changed in a database. There are many tools that can provide a list of before and after differences (e.g. redgate SQL Data Compare), but SAS PROC COMPARE can be coupled with other tools in base SAS to analyze the changes. This paper will explore using the output file produced by PROC COMPARE and the SAS Macro language to produce spreadsheets of detailed differences and summaries to perform this task.

INTRODUCTION

As data managers on an analysis team we were tasked with verifying that programming by two separate external coding teams was correctly implemented (without introducing errors to existing processing) for a series of test cases. The verification was extremely time consuming and needed to be repeated for each set of changes by both external teams. To reduce the time required and improve the accuracy of our verification, the PROC COMPARE procedure was used to determine and summarize the differences in all tables from before and after images of the project database. Later, we used the same techniques to compare images of the database before and after a series of mass updates. This paper describes the comparison techniques using a single table. The complete code for the example used in the paper is given in Appendix A.

PROC COMPARE OPTIONS AND RESULTS

PROC COMPARE provides the capability to determine the differences in two SAS data sets. Procedure options provide control over the types of differences displayed in reports or written to an output dataset. In this paper, we will make use the options listed below:

<u>BASE=</u>SAS-data-set
    (specifies the data set to use as the base data set.)
<u>COMPARE=</u>SAS-data-set
    (specifies the data set to use as the comparison data set.)
<u>NOPRINT</u>
    (suppresses all printed output.
    Tip:You may want to use this option when you are creating one or more output data sets.)
<u>OUT=</u>SAS-data-set
    (creates an output data set.)
<u>OUTBASE</u>
    (writes an observation for each observation in the base data set.)
<u>OUTCOMP</u>
    (writes an observation for each observation in the comparison data set)

<u>OUTDIF</u>
    (writes an observation to the output data set for each pair of matching observations.)

In addition the following statement is used

<u>ID Statement</u>
    (Identify variables to use to match observations).

In the example, we will be comparing before and after changes versions of a mock Visits data set using the unique combinations of variables Site, SubjectID, and VisitType. To do this the input data sets need to be sorted on these variables, and in PROC COMPARE the variables will be used on the ID statement. The following code segment accomplishes this:

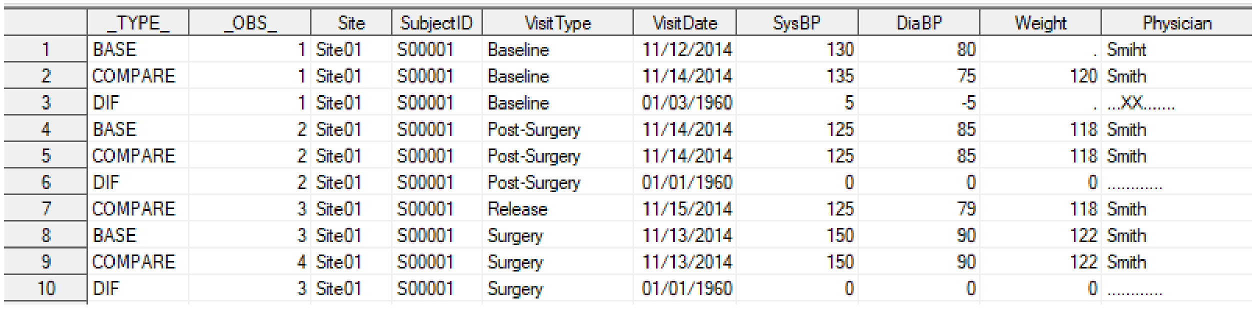
```
proc sort data=ORIGINAL.Visits;
  by Site SubjectID VisitType;
run;

proc sort data=Changed.Visits;
  by Site SubjectID VisitType;
run;


proc compare BASE=ORIGINAL.Visits
             COMPARE=CHANGED.Visits
             OUT=CompareOut_Visits
             outbase outcomp outdif noprint;
  title "Compare: Visits Table Column differences";
  ID Site SubjectID VisitType;
run;
```

Below is a description of what the code segment above accomplishes:

1. The ORIGINAL.Visits table is assigned to the base data set in the comparison
2. The CHANGED.Visits table is assigned to the compare data set.
3. The output of the PROC COMPARE will be written to the CompareOut_Visits data set.
4. CompareOut_Visits will contain one row for each observation in the base dataset, one row for each observation in the compare data set, and one row showing the differences between the matching row in base and compare.

   Note: if a row exists only in BASE= or only in COMPARE= there will be no difference row.
5. The ID statement shows that the comparison will be performed on rows in BASE= and COMPARE= data sets which match on Site, SubjectID, and VisitType.

Error! Reference source not found. **- OUT data set from PROC COMPARE.**



	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht
2	COMPARE	1	Site01	S00001	Baseline	11/14/2014	135	75	120	Smith
3	DIF	1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......
4	BASE	2	Site01	S00001	Post-Surgery	11/14/2014	125	85	118	Smith
5	COMPARE	2	Site01	S00001	Post-Surgery	11/14/2014	125	85	118	Smith
6	DIF	2	Site01	S00001	Post-Surgery	01/01/1960	0	0	0	....
7	COMPARE	3	Site01	S00001	Release	11/15/2014	125	79	118	Smith
8	BASE	3	Site01	S00001	Surgery	11/13/2014	150	90	122	Smith
9	COMPARE	4	Site01	S00001	Surgery	11/13/2014	150	90	122	Smith
10	DIF	3	Site01	S00001	Surgery	01/01/1960	0	0	0	....

Figure 1 shows the observations in the OUT data set for SubjectID S00001 from the example. Following is description of the contents of the data set.

The value of the _TYPE_ column indicates the source of the observation. BASE indicates the row is from the BASE data set, COMPARE indicates the row is from the COMPARE data set, and DIF indicates the row was generated by PROC COMPARE and contains the differences between the variables in BASE data set and COMPARE data set that match on the variables specified in the ID statement.

The _OBS_ column gives the observation number of the row in its source data set.

Note that for VisitType Release (row 7) there are no rows with _TYPE_ of BASE or DIF. This indicates that the ID variable combination for this row only exists in the COMPARE data set.

Row 3 in the Figure 1 shows the differences between the rows in BASE and COMPARE data sets for the ID variable values of Site=Site01, SubjectID=S00001, and VisitType=Baseline

> VisitDate has changed from 11/12/2014 to 11/14/2014. Since the VisitDate column is formatted as a date, the difference is also shown in date format. For date formatted variables 01/01/1960 indicates a difference of zero days.
>
> SysBP has changed from 130 to 135, a difference of positive 5
>
> DiaBP has changed from 80 to 75, a difference of negative 5
>
> Weight has changed from missing to 120, the missing value in the DIF row indicates a comparison could not be made. Note: PROC COMPARE options NOMISSBASE, NOMISSCOMP, NOMISSING can be used to modify how missing values are treated in comparisons. In this example we are using the default.
>
> Physican has changed from Smiht to Smith. For character variables '.' indicates that the strings in BASE and COMPARE were equal at the position, 'X' indicates that the character at the position were different.

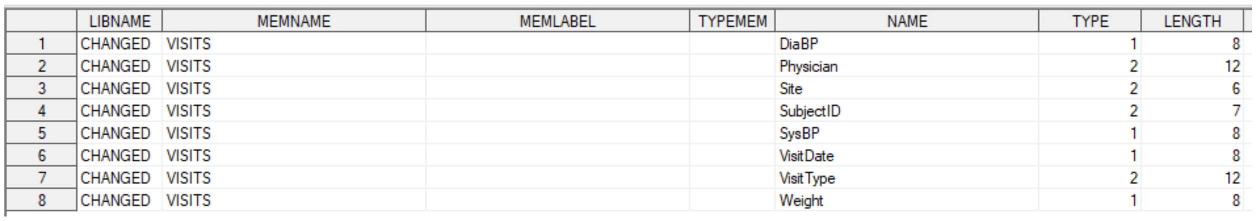
Row 6 in the Figure 1 is an example of a comparison where all variables were equal (zero for all numeric variables and no 'X's at any position for the character variables).

PROC CONTENTS OUT=FILE

Since differences for character and numeric variables are not presented in the same way by PROC COMPARE, our processing will need to take this into consideration. Fortunately the PROC CONTENTS procedure can be used to identify the variable type. The code below stores the PROC CONTENTS output for CHANGED.Visits into the data set Visits_Columns:

```
PROC CONTENTS NOPRINT DATA=CHANGED.Visits OUT=Visits_Columns;
run;
```

Figure 2 - OUT= file from PROC CONTENTS.



	LIBNAME	MEMNAME	MEMLABEL	TYPEMEM	NAME	TYPE	LENGTH
1	CHANGED	VISITS			DiaBP	1	8
2	CHANGED	VISITS			Physician	2	12
3	CHANGED	VISITS			Site	2	6
4	CHANGED	VISITS			SubjectID	2	7
5	CHANGED	VISITS			SysBP	1	8
6	CHANGED	VISITS			VisitDate	1	8
7	CHANGED	VISITS			VisitType	2	12
8	CHANGED	VISITS			Weight	1	8

Figure 2 shows part of the PROC CONTENTS output data set.

Things to note are:

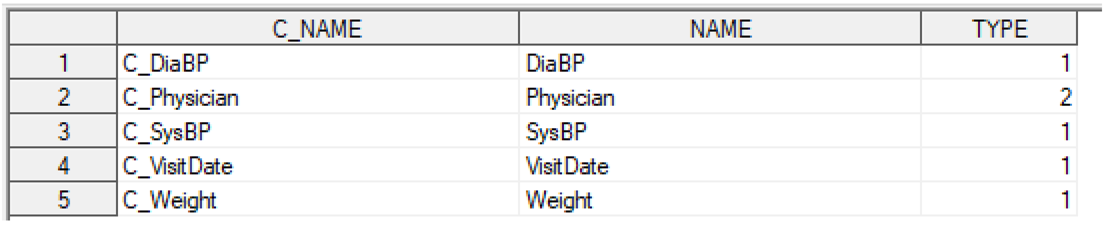
1. A value of 1 in the TYPE column indicates a numeric variable
2. A value of 2 in the TYPE column indicates a character variable.
3. Column NAME contains the variable names from CHANGED.Visits data set.

In subsequent processing, we will need to determine if there are differences in both the character and numeric variables for rows in the BASE and COMPARE data sets matching on the ID variables. The DATA step below starts the preparation to create macro variables which will hold the variable names being compared in the DIF row and a corresponding set of variables to hold a flag indicating whether there was a differences for the variable. The flag variable are prefixed with "C_".

```
data Visits_Columns_for_MacroVar ;
  length C_NAME $32;
  set Visits_Columns;
  /* ID variables don't need to be checked */
  where NAME not in ('Site','SubjectID','VisitType' );
  if length(NAME) le 30 then
    C_NAME=cat("C_",trim(NAME));
  else
    C_NAME=cat("C_",substr(NAME,1,30));
  keep C_NAME NAME TYPE;
run;
```

Figure 3 shows the output data set from the preceding DATA step.

Figure 3 - Visits_Columns_for_MacroVar



	C_NAME	NAME	TYPE
1	C_DiaBP	DiaBP	1
2	C_Physician	Physician	2
3	C_SysBP	SysBP	1
4	C_VisitDate	VisitDate	1
5	C_Weight	Weight	1

PROC SQL TO MACRO LANGUAGE INTERFACE

In the PROC SQL procedure step below, the SAS SQL to Macro Language interface is used to create macro variables to hold the character variable names, a variable indicating if the character value has changed, the numeric variable names, and a variable indicating if the numeric variable has changed:

```
proc sql noprint;
  select NAME, C_NAME
     into :CMPVARS_N separated by " " ,:CMPVARS_N_N  separated by " "
       from Visits_Columns_for_MacroVar where TYPE=1;/* numeric
variables */
  select NAME, C_NAME
     into :CMPVARS_C separated by " " ,:CMPVARS_C_N  separated by " "
       from Visits_Columns_for_MacroVar where TYPE=2;/* character
variables*/
quit;
```

The values of the MACRO variables created in the PROC SQL step above are shown below:

CMPVARS_N	DiaBP SysBP VisitDate Weight
CMPVARS_N_N	C_DiaBP C_SysBP C_VisitDate C_Weight
CMPVARS_C	Physician
CMPVARS_C_N	C_Physician

DATA STEP ARRAY PROCESSING FOR VARIABLE DIFFERENCES

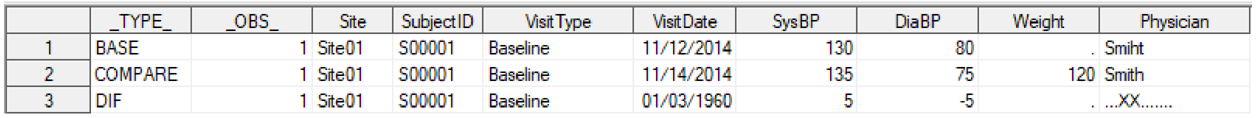
In the DATA step below the rows from the PROC COMPARE OUT data set with _TYPE_='DIF' are processed using the macro variables generated above. Two arrays for the numeric variables (one holding the numeric values from the DIF row and the second to hold a variable to indicate if the value of the variable has changed) are created from CMPVARS_N and CMPVARS_N_N. Similarly, two arrays are created for the character variables from CMPVARS_C and CMPVARS_C_N.

Note: at least one variable must have a changed value in order for an observation to be written to Delta_Visits.

```
data Deltas_Visits (drop=_TYPE_ _OBS_);
   set CompareOut_Visits (where=(_TYPE_="DIF"));
   /* array to hold numeric variables in DIF row */
   array changes_N(*) &CMPVARS_N;
   /* array of indicators for numeric variable changes */
   array changes_N_cnt(*) &CMPVARS_N_N;
   /* array to hold character variables in DIF row */
   array changes_C(*) &CMPVARS_C;
   /* array of indicators for character variable changes */
   array changes_C_cnt(*) &CMPVARS_C_N;
   * Process the numeric variables *;
   deltas_N=0;
   do i=1 to DIM(changes_N);
      /* if the value from the DIFF row <> 0, there is a change */
      if changes_N(i) ne 0 then do;
```

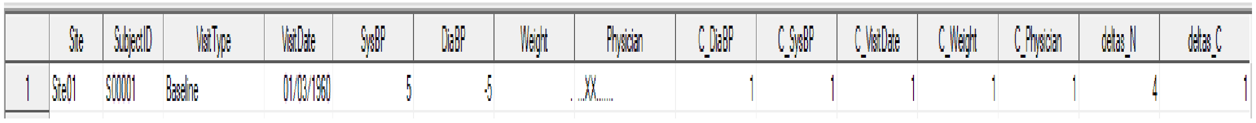
```
         /* when there is a change update a counter and set the
           indicator variable to 1*/
         deltas_N+1; changes_N_cnt(i)=1;
       end;
        /* when there is no change, set the indicator variable to 0 */
       else changes_N_cnt(i)=0;
    end;
    * Process the character variables *;
    deltas_C=0;
    do i=1 to DIM(changes_C);
      /*if the value from the DIF row contains an X, there is a change*/
      if index(changes_C(i),"X") > 0 then do;
        /* when there is a change update a counter and set the indicator
          variable to 1*/
        deltas_C+1; changes_C_cnt(i)=1;
      end;
        /* when there is no change, set the indicator variable to 0 */
      else changes_C_cnt(i)=0;
   end;
   /* only output an observation if there is at least one difference */
   if sum(deltas_N,deltas_C) > 0;
run;
```

Figure 4.a. - Rows in COMPARE OUT data set for Site=Site01, SubjectID=S0001, and VisitType=Baseline ID variable combination



	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht
2	COMPARE	1	Site01	S00001	Baseline	11/14/2014	135	75	120	Smith
3	DIF	1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......

Figure 4.b. - Deltas_Visits row showing variables flagged as different for ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician	C_DiaBP	C_SysBP	C_VisitDate	C_Weight	C_Physician	deltas_N	deltas_C
1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......	1	1	1	1	1	4	1

Figure 4.a. shows the rows from the PROC COMPARE OUT data set for the Site=Site01, SubjectID=S0001, and VisitType=Baseline ID combination. Please note that all of the non-ID variables have changed. Figure 4.b. shows that the Deltas_Visits data set for this ID combination has the change indicator variables (C_*) correctly set to reflect this.

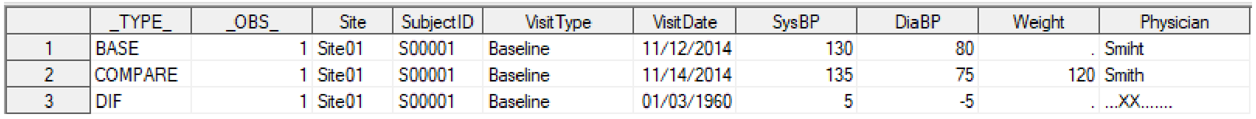
SELECT THE BEFORE AND AFTER VALUES FOR CHANGES

Now we will select the pre-values and post values for the ID combinations where a variable has changed. To select the pre-values, we will join the rows from the PROC COMPARE OUT data set where the rows have _TYPE_=BASE with the Delta_Visits on the ID variables. This will give us all rows in the BASE table which have differences along with their original values.

```
proc sql;
  create table Visits_prior_vals as
  select cmp.*, dlta.*
```

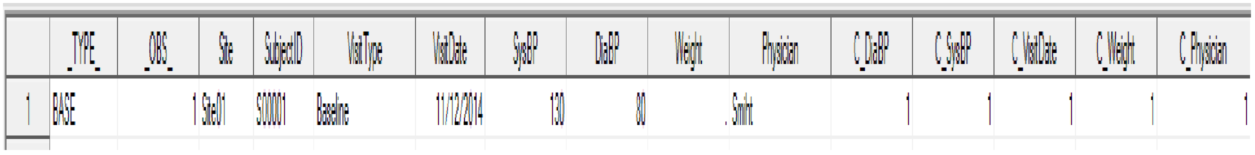
```
        from CompareOUT_Visits as cmp, deltas_visits (keep=Site SubjectID
        VisitType c_:) as dlta
         where  cmp.Site=dlta.Site and cmp.SubjectID=dlta.SubjectID and
  cmp.VisitType=dlta.VisitType and cmp._TYPE_='BASE';;
  quit;
```

Figure 5.a. - Rows in COMPARE OUT data set for Site=Site01, SubjectID=S0001, and VisitType=Baseline ID variable combination



	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht
2	COMPARE	1	Site01	S00001	Baseline	11/14/2014	135	75	120	Smith
3	DIF	1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......

Figure 5.b. - Visits_prior_vals row showing variables flagged as different for ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



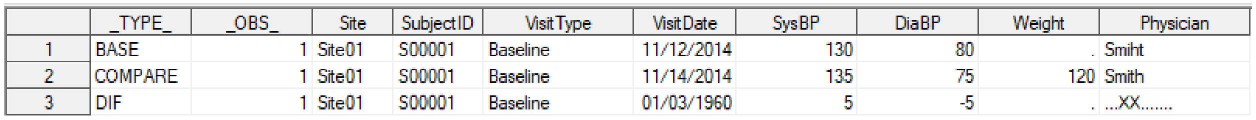
	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician	C_DiaBP	C_SysBP	C_VisitDate	C_Weight	C_Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht	1	1	1	1	1

Figure 5.a. shows the rows from the PROC COMPARE OUT data set for the Site=Site01, SubjectID=S0001, and VisitType=Baseline ID combination. Figure 5.b. shows that the Visits_prior_vals BASE values for this ID combination have been correctly selected. At this point the rows in the Visits_prior_vals contain the pre-values along with the corresponding indicator (C_*) variables to identify changes.

The following DATA step checks the indicator variables and will output a row containing the variable name and its pre-change value if its corresponding indicator variable equals a value of 1:

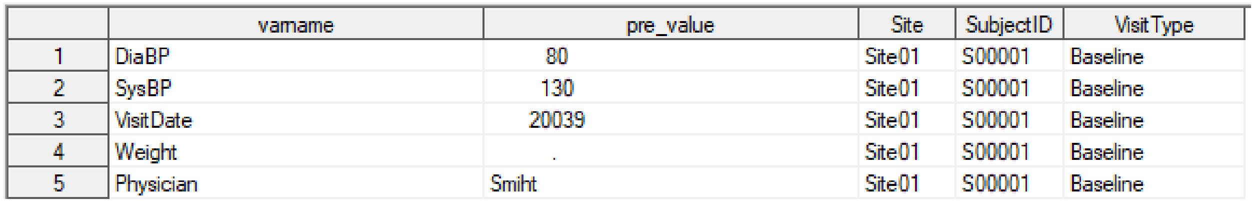
```
  data Visits_bv;
    length varname $ 32 pre_value $ 64;
    set Visits_prior_vals;
   /* array to hold numeric variables in BASE row */
    array changes_N(*) &CMPVARS_N;
   /* array of indicators for numeric variable changes */
    array changes_N_cnt(*) &CMPVARS_N_N;
   /* array to hold character variables in BASE row */
    array changes_C(*) &CMPVARS_C;
   /* array of indicators for character variable changes */

    array changes_C_cnt(*) &CMPVARS_C_N;
   *Process the numeric variables*;
    do i=1 to DIM(changes_N);
       /* if the indicator shows this variable has changed */
      if changes_N_cnt(i) = 1 then do;
         /* use the vname function to obtain the variable name */
         varname=vname(changes_N(i));
         /* place the formatted numeric value into pre_value */
         pre_value=put(changes_N(i),best.);
         /* output a row for every variable involved in a change */
        output;
      end;
    end;
    *Process the character variables;
    do i=1 to DIM(changes_C);
       /* if the indicator shows this variable has changed */
       if changes_C_cnt(i) = 1 then do;
          /* use the vname function to obtain the variable name */
          varname=vname(changes_C(i));
          /* place the character value pre_value */
          pre_value=changes_C(i);
          /* output a row for every variable involved in a change */
          output;
      end;
    end;

    keep Site SubjectID VisitType varname pre_value;
  run;
```

Figure 6.a. - Rows in COMPARE OUT data set for Site=Site01, SubjectID=S0001, and VisitType=Baseline ID variable combination



	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht
2	COMPARE	1	Site01	S00001	Baseline	11/14/2014	135	75	120	Smith
3	DIF	1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......

Figure 6.b. - Visits_bv showing the before values for the ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



	varname	pre_value	Site	SubjectID	VisitType
1	DiaBP	80	Site01	S00001	Baseline
2	SysBP	130	Site01	S00001	Baseline
3	VisitDate	20039	Site01	S00001	Baseline
4	Weight	.	Site01	S00001	Baseline
5	Physician	Smiht	Site01	S00001	Baseline

Figure 6.a. shows the rows from the PROC COMPARE OUT data set for the Site=Site01, SubjectID=S0001, and VisitType=Baseline ID combination. Figure 6.b. shows that the Visits_bv data set contains the variable names and pre-change values for variables that underwent a change.

At this point we have the variables/values from the BASE data set that are involved in differences each on their own row (20039 is 11/12/2014 when formatted as a date).

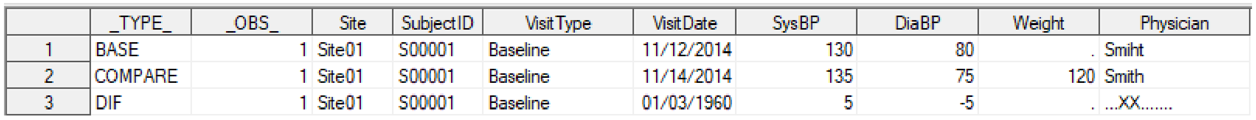
The post change values can be found using the same method – starting by selecting the _TYPE_=COMPARE rows from the PROC COMPARE OUT data set as shown the PROC SQL step below:

```
proc sql;
  create table Visits_post_vals as
  select cmp.*, dlta.*
      from CompareOUT_Visits as cmp, deltas_visits
      (keep=Site SubjectID  VisitType c_:) as dlta
       where  cmp.Site=dlta.Site and cmp.SubjectID=dlta.SubjectID and
cmp.VisitType=dlta.VisitType and cmp._TYPE_='COMPARE';
quit;
```

The full process can be found in Appendix A. The data set Visits_av contains the post change values at its completion.

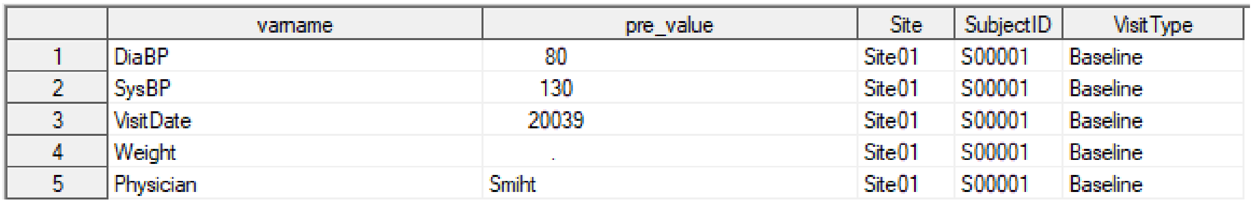
Figure 7.a shows the rows from the PROC COMPARE OUT data set for the Site=Site01, SubjectID=S0001, and VisitType=Baseline ID variable combination. Figure 7.b. gives the before values for this ID variable combination from the Visits_bv data set and Figure 7.c. gives the after values from the Visits_av data set.

Figure 7.a. - Rows in COMPARE OUT data set for Site=Site01, SubjectID=S0001, and VisitType=Baseline ID variable combination



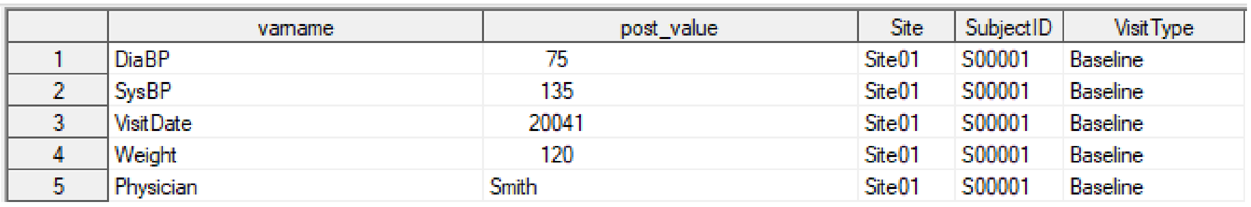
	TYPE	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
1	BASE	1	Site01	S00001	Baseline	11/12/2014	130	80	.	Smiht
2	COMPARE	1	Site01	S00001	Baseline	11/14/2014	135	75	120	Smith
3	DIF	1	Site01	S00001	Baseline	01/03/1960	5	-5	.	...XX.......

Figure 7.b. - Visits_bv data set showing the before values for the ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



	varname	pre_value	Site	SubjectID	VisitType
1	DiaBP	80	Site01	S00001	Baseline
2	SysBP	130	Site01	S00001	Baseline
3	VisitDate	20039	Site01	S00001	Baseline
4	Weight	.	Site01	S00001	Baseline
5	Physician	Smiht	Site01	S00001	Baseline

Figure 7.c. - Visits_av data set showing the after values for the ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



	varname	post_value	Site	SubjectID	VisitType
1	DiaBP	75	Site01	S00001	Baseline
2	SysBP	135	Site01	S00001	Baseline
3	VisitDate	20041	Site01	S00001	Baseline
4	Weight	120	Site01	S00001	Baseline
5	Physician	Smith	Site01	S00001	Baseline

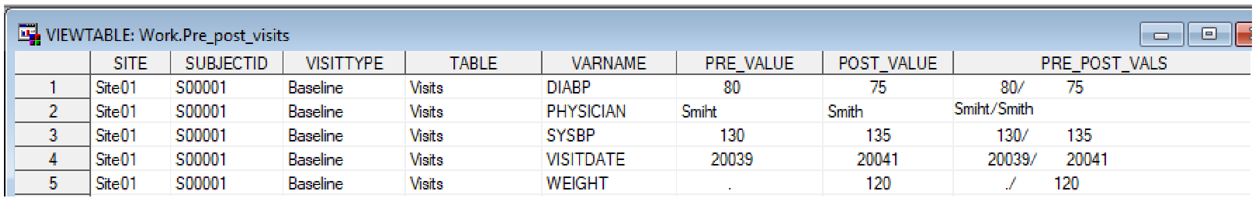
REPORTING ON CHANGES

At this point, Visits_bv and Visits_av data sets can be merged on the ID variables plus the varname column allowing analyses to be performed on differences in pre_value and post_value variables. The code below creates the data set pre_post_Visits which will be the basis for reporting on changes.

```
proc sort data=Visits_bv;
  by Site SubjectID VisitType varname ;
run;

proc sort data=Visits_av;
  by Site SubjectID VisitType varname ;
run;

data pre_post_Visits;
   retain Site SubjectID VisitType table varname pre_value post_value;
   length table $32 pre_post_vals $ 128;
   merge Visits_bv (in=b) Visits_av (in=a);
```

```
     by Site SubjectID VisitType varname ;
  table="Visits";
  pre_post_vals=cat(trim(pre_value),"/",trim(post_value));
 run;
```

Figure 8 - pre_post_Visits changed variables and their pre/post values for the ID variable combination Site=Site01, SubjectID=S0001, and VisitType=Baseline



VIEWTABLE: Work.Pre_post_visits

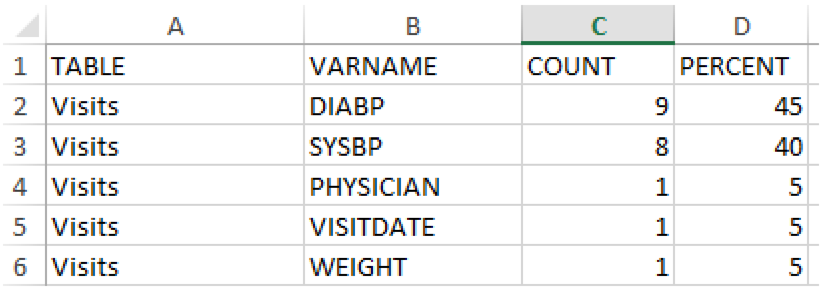
	SITE	SUBJECTID	VISITTYPE	TABLE	VARNAME	PRE_VALUE	POST_VALUE	PRE_POST_VALS
1	Site01	S00001	Baseline	Visits	DIABP	80	75	80/ 75
2	Site01	S00001	Baseline	Visits	PHYSICIAN	Smiht	Smith	Smiht/Smith
3	Site01	S00001	Baseline	Visits	SYSBP	130	135	130/ 135
4	Site01	S00001	Baseline	Visits	VISITDATE	20039	20041	20039/ 20041
5	Site01	S00001	Baseline	Visits	WEIGHT	.	120	./ 120

We now have the changed values in a format that can be used for analysis and reporting. Exporting the output of PROC FREQ into Microsoft Excel will provide examples of differences that might give some insight into the changes that have occurred.

The first report will be a look at the frequency of changes by variable name:

```
proc sort data = pre_post_Visits;
  by table varname;
run;

proc freq noprint data=pre_post_Visits order=freq;
  by table ;
  table varname / out=pre_post_Visits_vars_freq;
run;

PROC EXPORT
     DATA= WORK.Pre_post_visits_vars_freq
     OUTFILE= "H:\MWSUG Fall 2018\Reports\Count of Changes by Varname.xlsx"
     DBMS=EXCEL REPLACE;
     SHEET="Differences";
RUN;
```

Figure 9 - Counts of changes by VARNAME



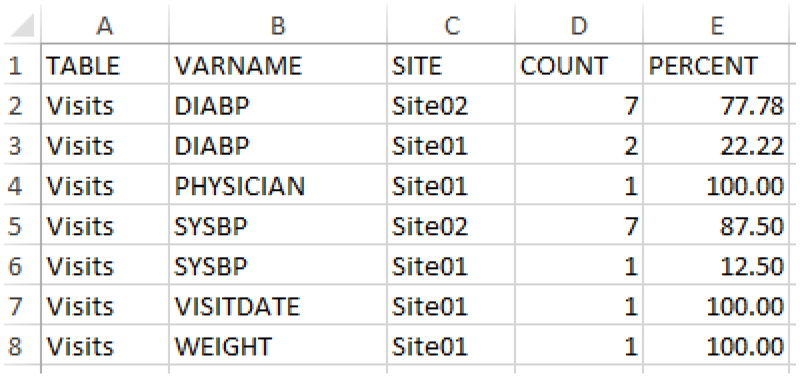
	A	B	C	D
1	TABLE	VARNAME	COUNT	PERCENT
2	Visits	DIABP	9	45
3	Visits	SYSBP	8	40
4	Visits	PHYSICIAN	1	5
5	Visits	VISITDATE	1	5
6	Visits	WEIGHT	1	5

Figure 9 gives the spreadsheet generated by the export of the PROC FREQ output. DIABP and SYSBP have a lot of differences compared to the other variables.

In order to drill down another layer, the code below performs a frequency of VARNAME by Site:

```
proc sort data = pre_post_Visits;
  by table varname Site;
run;

proc freq noprint data=pre_post_Visits order=freq;
  by table varname;
  table Site / out=pre_post_Visits_sitevars_freq;
run;

PROC EXPORT
   DATA= WORK.pre_post_Visits_sitevars_freq
   OUTFILE= "H:\MWSUG Fall 2018\Reports\Count of Changes by Varname and
Site.xlsx"
   DBMS=EXCEL REPLACE;
   SHEET="Differences";
RUN;
```

Figure 10 - Counts of changes by VARNAME and Site



	A	B	C	D	E
1	TABLE	VARNAME	SITE	COUNT	PERCENT
2	Visits	DIABP	Site02	7	77.78
3	Visits	DIABP	Site01	2	22.22
4	Visits	PHYSICIAN	Site01	1	100.00
5	Visits	SYSBP	Site02	7	87.50
6	Visits	SYSBP	Site01	1	12.50
7	Visits	VISITDATE	Site01	1	100.00
8	Visits	WEIGHT	Site01	1	100.00

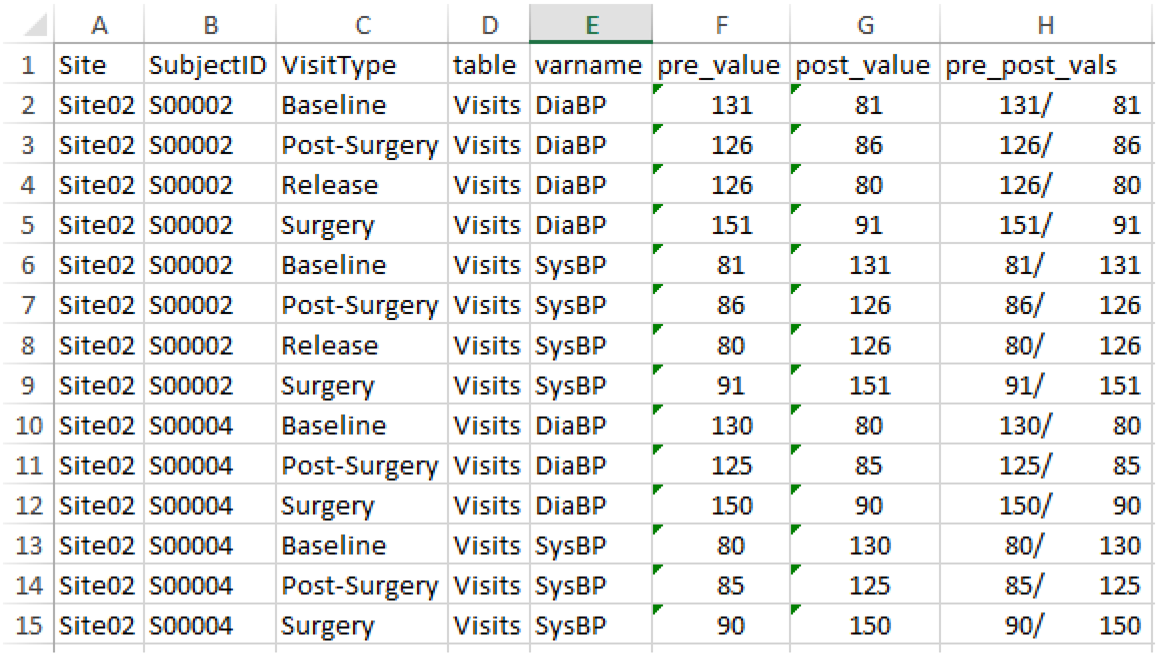
Figure 10 gives the spreadsheet generated by the export of the PROC FREQ and we can see that DIABP and SYSDP have had a lot of changes at Site02.

The code below will report on the individual change detail, including the pre and post values:

```
proc sort data=pre_post_Visits;
  by Site SubjectID varname;
run;

PROC EXPORT DATA= WORK.pre_post_Visits (where=(Site='Site02' and Varname in
('DiaBP','SysBP')))
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Site02_DiaBP_SysBP.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="Differences";
RUN;
```

Figure 11 lets us see that the SysBP and DiaBP values were originally switched and the differences are due to corrections.

Figure 11 - Site02 changes in SysBP and DiaBP



	A	B	C	D	E	F	G	H
1	Site	SubjectID	VisitType	table	varname	pre_value	post_value	pre_post_vals
2	Site02	S00002	Baseline	Visits	DiaBP	131	81	131/ 81
3	Site02	S00002	Post-Surgery	Visits	DiaBP	126	86	126/ 86
4	Site02	S00002	Release	Visits	DiaBP	126	80	126/ 80
5	Site02	S00002	Surgery	Visits	DiaBP	151	91	151/ 91
6	Site02	S00002	Baseline	Visits	SysBP	81	131	81/ 131
7	Site02	S00002	Post-Surgery	Visits	SysBP	86	126	86/ 126
8	Site02	S00002	Release	Visits	SysBP	80	126	80/ 126
9	Site02	S00002	Surgery	Visits	SysBP	91	151	91/ 151
10	Site02	S00004	Baseline	Visits	DiaBP	130	80	130/ 80
11	Site02	S00004	Post-Surgery	Visits	DiaBP	125	85	125/ 85
12	Site02	S00004	Surgery	Visits	DiaBP	150	90	150/ 90
13	Site02	S00004	Baseline	Visits	SysBP	80	130	80/ 130
14	Site02	S00004	Post-Surgery	Visits	SysBP	85	125	85/ 125
15	Site02	S00004	Surgery	Visits	SysBP	90	150	90/ 150

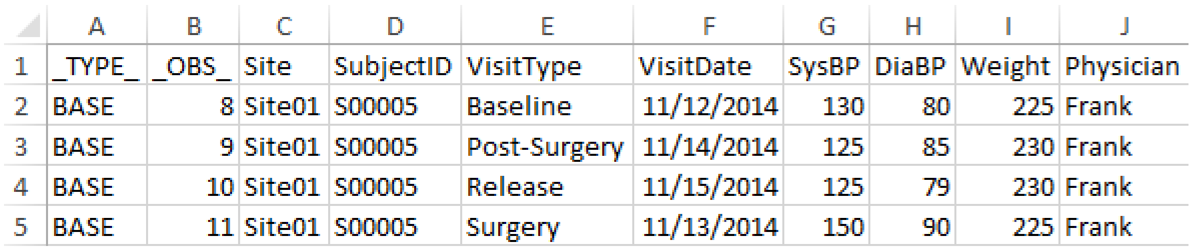
At this point the changes to rows where there was a match on the ID variables in PROC COMPARE have been identified. To complete the review we need to examine the cases where there was no match on the ID variables.

The following PROC SQL step creates a data set containing the rows only in the BASE data set as well as a data set containing the rows only in the COMPARE data set. The two data sets are then placed into a spreadsheets by PROC EXPORT.

```
proc sql;
create table visits_base_only (drop=key) as
  select *, cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
    from CompareOut_Visits
    where _TYPE_='BASE' and
          cat(trim(Site),trim(SubjectID),trim(VisitType)) not in
      (select cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
         from CompareOut_Visits
         where _type_='DIF')
    order by key;
create table visits_compare_only (drop=key) as
  select *, cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
     from CompareOut_Visits
     where _TYPE_='COMPARE' and
          cat(trim(Site),trim(SubjectID),trim(VisitType)) not in
       (select cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
          from CompareOut_Visits
          where _type_='DIF')
  order by key;
quit;

PROC EXPORT DATA= WORK.visits_base_only
     OUTFILE= "H:\MWSUG Fall 2018\Reports\Rows_only_in_the_BASE_Dataset.xlsx"
     DBMS=EXCEL REPLACE;
     SHEET="Base only Rows";
RUN;


PROC EXPORT DATA= WORK.visits_compare_only
  OUTFILE= "H:\MWSUG Fall 2018\Reports\Rows_only_in_the_COMPARE_Dataset.xlsx"
  DBMS=EXCEL REPLACE;
  SHEET="Compare only Rows";
RUN;
```

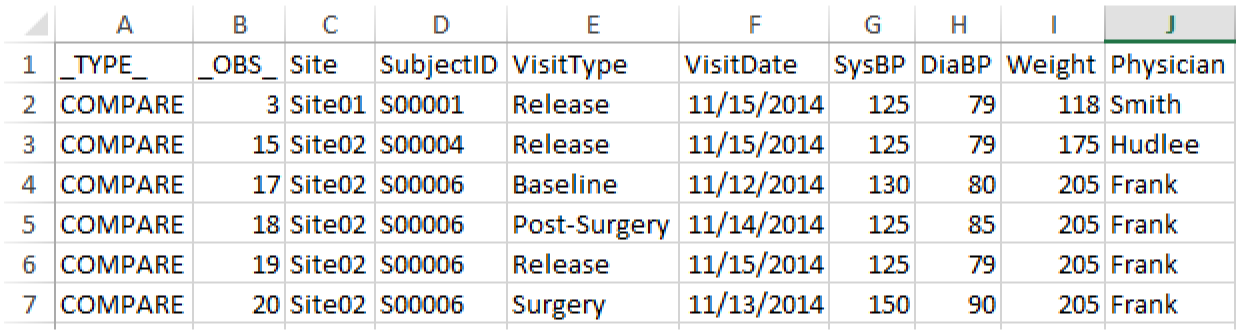
Figures 12 and 13 give the results of the PROC EXPORTS for the BASE only and COMPARE only rows

Figure 12 - Rows only in the BASE data set



	A	B	C	D	E	F	G	H	I	J
1	_TYPE_	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
2	BASE	8	Site01	S00005	Baseline	11/12/2014	130	80	225	Frank
3	BASE	9	Site01	S00005	Post-Surgery	11/14/2014	125	85	230	Frank
4	BASE	10	Site01	S00005	Release	11/15/2014	125	79	230	Frank
5	BASE	11	Site01	S00005	Surgery	11/13/2014	150	90	225	Frank

Figure 13 - Rows only in the COMPARE data set



	A	B	C	D	E	F	G	H	I	J
1	_TYPE_	_OBS_	Site	SubjectID	VisitType	VisitDate	SysBP	DiaBP	Weight	Physician
2	COMPARE	3	Site01	S00001	Release	11/15/2014	125	79	118	Smith
3	COMPARE	15	Site02	S00004	Release	11/15/2014	125	79	175	Hudlee
4	COMPARE	17	Site02	S00006	Baseline	11/12/2014	130	80	205	Frank
5	COMPARE	18	Site02	S00006	Post-Surgery	11/14/2014	125	85	205	Frank
6	COMPARE	19	Site02	S00006	Release	11/15/2014	125	79	205	Frank
7	COMPARE	20	Site02	S00006	Surgery	11/13/2014	150	90	205	Frank

CONCLUSION

Through the use of the PROC COMPARE OUT dataset, the PROC SQL to SAS macro interface, and DATA step array programming we were able to create solution to meet our need to improve the speed and accuracy of our validation effort. In addition we developed parameterized SAS Macros that leveraged the techniques presented here to identify and report on the differences in an entire database in small fraction of the time we were spending on manually determining the differences.

ACKNOWLEDGMENTS

I would like to thank Anna Kispert for her work in helping to develop and streamline the code for this project. I would also like to thank Janetta Matesan and Bob Davis for allowing me to present this work.

CONTACT INFORMATION

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

Larry Riggen
Indiana University Department of Biostatistics
ldriggen@iu.edu

APPENDIX A

```
* Program name:   MWSUG 2018 Paper SP-069 Sammple Code.sas
* Author:         Larry Riggen
* Creation Date:  August 2018
* Purpose:        Demonstrate the use of PROC COMPARE, the SAS SQL to SAS Macro interface and
*                 array processing to create a dataset of differences in two tables structured
*                 to allow analysis
*
* Changes:
;

libname ORIGINAL "H:\MWSUG Fall 2018\Data\Original";
libname CHANGED  "H:\MWSUG Fall 2018\Data\Changed";


* Create an original version of a mock visits table.;


data ORIGINAL.Visits;
  input @01 SubjectID $7.
        @08 VisitType $12.
               @21 VisitDate MMDDYY10.
        @33 SysBP     3.
           @37 DiaBP     3.
               @41     Weight    3.
               @45 Site         $6.
               @52 Physician $12.;

  format VisitDate MMDDYY10.;
datalines;
S00001 Baseline     11/12/2014  130  80     Site01 Smiht
S00001 Surgery      11/13/2014  150  90 122 Site01 Smith
S00001 Post-Surgery 11/14/2014  125  85 118 Site01 Smith
S00002 Baseline     11/12/2014   81 131 145 Site02 Jones
S00002 Surgery      11/13/2014   91 151 145 Site02 Jones
S00002 Post-Surgery 11/14/2014   86 126 145 Site02 Jones
S00002 Release      11/15/2014   80 126 145 Site02 Jones
S00003 Baseline     11/12/2014  130  80 180 Site01 Palakal
S00003 Surgery      11/13/2014  150  90 180 Site01 Palakal
S00003 Post-Surgery 11/14/2014  125  85 180 Site01 Palakal
S00003 Release      11/15/2014  125  79 180 Site01 Palakal
S00004 Baseline     11/12/2014   80 130 175 Site02 Hudlee
S00004 Surgery      11/13/2014   90 150 175 Site02 Hudlee
S00004 Post-Surgery 11/14/2014   85 125 175 Site02 Hudlee
S00005 Baseline     11/12/2014  130  80 225 Site01 Frank
S00005 Surgery      11/13/2014  150  90 225 Site01 Frank
S00005 Post-Surgery 11/14/2014  125  85 230 Site01 Frank
S00005 Release      11/15/2014  125  79 230 Site01 Frank
;


* Create a changed version of a mock visits table.;

data CHANGED.Visits;
  input @01 SubjectID $7.
        @08 VisitType $12.
               @21 VisitDate MMDDYY10.
        @33 SysBP     3.
           @37 DiaBP     3.
               @41     Weight    3.
               @45 Site         $6.
               @52 Physician $12.;
  format VisitDate MMDDYY10.;
datalines;
S00001 Baseline     11/14/2014  135  75 120 Site01 Smith
S00001 Surgery      11/13/2014  150  90 122 Site01 Smith
S00001 Post-Surgery 11/14/2014  125  85 118 Site01 Smith
S00001 Release      11/15/2014  125  79 118 Site01 Smith
S00002 Baseline     11/12/2014  131  81 145 Site02 Jones
S00002 Surgery      11/13/2014  151  91 145 Site02 Jones
S00002 Post-Surgery 11/14/2014  126  86 145 Site02 Jones
S00002 Release      11/15/2014  126  80 145 Site02 Jones
S00003 Baseline     11/12/2014  130  80 180 Site01 Palakal
S00003 Surgery      11/13/2014  150  90 180 Site01 Palakal
S00003 Post-Surgery 11/14/2014  125  90 180 Site01 Palakal
S00003 Release      11/15/2014  125  79 180 Site01 Palakal
S00004 Baseline     11/12/2014  130  80 175 Site02 Hudlee
S00004 Surgery      11/13/2014  150  90 175 Site02 Hudlee
S00004 Post-Surgery 11/14/2014  125  85 175 Site02 Hudlee
S00004 Release      11/15/2014  125  79 175 Site02 Hudlee
S00006 Baseline     11/12/2014  130  80 205 Site02 Frank
S00006 Surgery      11/13/2014  150  90 205 Site02 Frank
S00006 Post-Surgery 11/14/2014  125  85 205 Site02 Frank
S00006 Release      11/15/2014  125  79 205 Site02 Frank
;


* The comparison will be done using Site, SubjectID, and
* VisitType as the ID variables. Sort the original and
* changed datasets in preparation for the comparison;


proc sort data=ORIGINAL.Visits;
  by Site SubjectID VisitType;
run;

proc sort data=Changed.Visits;
  by Site SubjectID VisitType;
run;


* Run PROC COMPARE and generate the output dataset CompareOut_Visits.
* CompareOut_Visits will contain all rows from the BASE dataset, all rows
* from the COMPARE dataset, and a difference row for cases where there
* is a match between the ID variables in BASE and COMPARE;

proc compare BASE=ORIGINAL.Visits
             COMPARE=CHANGED.Visits
                        OUT=CompareOut_Visits
                        outbase outcomp outdif noprint;
  title "Compare: Visits Table Column differences";
  ID Site SubjectID VisitType;
run;

* Write the output from PROC CONTENTS for the dataset  CHANGED-Visits
* to the file Visits_Columns.;

PROC CONTENTS NOPRINT ORDER=VARNUM DATA=CHANGED.Visits OUT=Visits_Columns;
run;


* Create variable C_NAME to correspond with the NAME variable from PROC CONTENTS *;
* The C_NAME variable will be used to indicate if there are changes in the NAME
* variable.;

data Visits_Columns_for_MacroVar ;
  length C_NAME $32;
  set Visits_Columns;
  where NAME not in ('Site','SubjectID','VisitType' ); /* ID variables don't need to be checked
*/
  if length(NAME) le 30 then
```

```
    C_NAME=cat("C_",trim(NAME));
  else
    C_NAME=cat("C_",substr(NAME,1,30));
  keep C_NAME NAME TYPE;
run;

* Use the SAS SQL to Macro language to create two sets of Macro variables
* (one set for numeric variables and one set for character variables).
* PROC COMPARE represents the changes in character and numeric variables differently
* and we will need to process them differently as well.
*
*              Macro variable CMPVARS_N contains the names of the numeric variables
*              Macro variable CMPVARS_N_N contains a list of indicator variables (for changes);
*              Macro variable CMPVARS_C contains the names of the character variables
*              Macro variable CMPVARS_C_N contains a list of indicator variables (for changes);

proc sql noprint;
  select NAME, C_NAME
     into :CMPVARS_N separated by " " ,:CMPVARS_N_N  separated by " "
        from Visits_Columns_for_MacroVar where TYPE=1; /* numeric variables */
  select NAME, C_NAME
     into :CMPVARS C separated by " " ,:CMPVARS C N  separated by " "
        from Visits_Columns_for_MacroVar where TYPE=2; /* character variables */
quit;


* Use the Macro variable create above to create arrays for the character and
* numeric variables and their change indicator variables.
* Note: only ID combinations  where there is a least one change are output to Delta_Visits;

 data Deltas_Visits (drop=_TYPE_ _OBS_ i);
  set CompareOut_Visits (where=(_TYPE_="DIF"));
  array changes_N(*) &CMPVARS_N;        /* array to hold numeric variables in DIF row */
  array changes_N_cnt(*) &CMPVARS_N_N;  /* array of indicators for numeric variable changes */
  array changes_C(*) &CMPVARS_C;        /* array to hold character variables in DIF row */
  array changes_C_cnt(*) &CMPVARS_C_N;  /* array of indicators for character variable changes */

  deltas_N=0;
  do i=1 to DIM(changes_N);
     if changes_N(i) ne 0 then do;     /* if the value from the DIFF row <> 0, there is a change
*/
     deltas_N+1; changes_N_cnt(i)=1; /* when there is a change update a counter and set the
                                     indicator variable to 1*/
     end;
        else changes_N_cnt(i)=0;     /* when there is no change, set the indicator variable to 0
                                     */
  end;
  * Process the character variables *;
  deltas_C=0;
  do i=1 to DIM(changes_C);
     if index(changes_C(i),"X") > 0 then do; /* if the value from the DIFF row contains an X,
                                              there is a change */
       deltas_C+1; changes_C_cnt(i)=1;       /* when there is a change update a counter and set
                                              the indicator variable to 1*/
     end;
        else changes_C_cnt(i)=0;              /* when there is no change, set the indicator
                                               variable to 0 */
  end;
  if sum(deltas_N,deltas_C) > 0;             /* only output an observation if there is at least
                                              one difference */
run;

* Find the before values for ID combinations with a difference;

* Select the rows from BASE in the PROC COMPARE output dataset that were found
* to have at least one difference. Include the character and numeric variables
* along with their change indicator variable from Delta_Visits;

proc sql;
  create table Visits_prior_vals as
  select cmp.*, dlta.*
```

```
      from CompareOUT_Visits as cmp, Deltas_Visits (keep=Site SubjectID VisitType c_:) as dlta
         where  cmp.Site=dlta.Site and cmp.SubjectID=dlta.SubjectID and
cmp.VisitType=dlta.VisitType and cmp._TYPE_='BASE';;
quit;

* If an indicator variable is flagged as a change (value of 1), output the
* corresponding variable and value to the Visits_bv dataset *;

data Visits_bv;
  length varname $ 32 pre_value $ 64;
  set Visits_prior_vals;
  array changes_N(*) &CMPVARS_N;         /* array to hold numeric variables in BASE row */
  array changes_N_cnt(*) &CMPVARS_N_N;  /* array of indicators for numeric variable changes */
  array changes_C(*) &CMPVARS_C;         /* array to hold character variables in BASE row */
  array changes_C_cnt(*) &CMPVARS_C_N;  /* array of indicators for character variable changes */
  *Process the numeric variables*;
  do i=1 to DIM(changes_N);
     if changes_N_cnt(i) = 1 then do;    /* if the indicator shows this variable has changed */
              varname=vname(changes_N(i));  /* use the vname function to obtain the variable
                                              name */
              pre_value=put(changes_N(i),best.);  /* place the formatted numeric value into
                                                     pre_value */
              output;                /* output a row for every variable involved in a change */
        end;
  end;
  *Process the character variables;
  do i=1 to DIM(changes_C);
     if changes_C_cnt(i) = 1 then do;   /*If the indicator shows this variable has changed */
              varname=vname(changes_C(i));  /* use the vname function to obtain the variable
                                            name */
              pre_value=changes_C(i);       /* place the character value pre_value */
              output;                /* output a row for every variable involved in a change */
        end;
  end;

  keep Site SubjectID VisitType varname pre_value;
run;

* Find the after values for ID combinations with a difference;

* Select the rows from COMPARE in the PROC COMPARE output dataset that were found
* to have at least one difference. Include the character and numeric variables
* along with their change indicator variable from Delta_Visits;

proc sql;
  create table Visits_post_vals as
  select cmp.*, dlta.*
     from CompareOUT_Visits as cmp, deltas_visits (keep=Site SubjectID VisitType c_:) as dlta
         where  cmp.Site=dlta.Site and cmp.SubjectID=dlta.SubjectID and
cmp.VisitType=dlta.VisitType and cmp._TYPE_='COMPARE';;
quit;

* If an indicator variable is flagged as a change (value of 1), output the
* corresponding variable and value to the Visits_bv dataset *;

data Visits_av;
  length varname $ 32 post_value $ 64;
  set Visits_post_vals;
  array changes_N(*) &CMPVARS_N; /* array to hold numeric variables in COMP row */
  array changes_N_cnt(*) &CMPVARS_N_N; /* array of indicators for numeric variable changes */
  array changes_C(*) &CMPVARS_C; /* array to hold character variables in COMP row */
  array changes_C_cnt(*) &CMPVARS_C_N; /* array of indicators for character variable changes */
  *Process the numeric variables*;
  do i=1 to DIM(changes_N);
     if changes_N_cnt(i) = 1 then do; /* if the indicator shows this variable has changed */
              varname=vname(changes_N(i));  /* use the vname function to obtain the variable
                                            name */
               post_value=put(changes_N(i),best.);  /* place the formatted numeric value into
                                                     pre_value */
              output; /* output a row for every variable involved in a change */
        end;
  end;
  *Process the character variables;
  do i=1 to DIM(changes_C);
     if changes_C_cnt(i) = 1 then do; /* if the indicator shows this variable has changed */
               varname=vname(changes_C(i));  /* use the vname function to obtain the variable
                                               name */
               post_value=changes_C(i);      /* place the character value pre_value */
               output;                /* output a row for every variable involved in a change */
        end;
  end;

  keep Site SubjectID VisitType varname post_value;
run;

* In preparation for merging the before and after values sort the
* before and after value datasets;

proc sort data=Visits_bv;
  by Site SubjectID VisitType varname ;
run;

proc sort data=Visits_av;
  by Site SubjectID VisitType varname ;
run;

* Merge the before and after datasets, creating a new variable that contains
* both the pre and post values.;

data pre_post_Visits;
   retain Site SubjectID VisitType table varname pre_value post_value;
   length table $32 pre_post_vals $ 128;
   merge Visits_bv (in=b) Visits_av (in=a);
     by Site SubjectID VisitType varname ;
  table="Visits";
  pre_post_vals=cat(trim(pre_value),"/",trim(post_value));
 run;

* The report will show the frequency of changes by varname ;

proc sort data = pre_post_Visits;
  by table varname;
run;

proc freq noprint data=pre_post_Visits order=freq;
  by table ;
  table varname / out=pre_post_Visits_vars_freq;
run;

PROC EXPORT DATA= WORK.Pre_post_visits_vars_freq
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Count of Changes by Varname.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="Differences";
RUN;

* The report will show the frequency of changes by varname and site;

proc sort data = pre_post_Visits;
  by table varname Site;
run;

proc freq noprint data=pre_post_Visits order=freq;
  by table varname;
  table Site / out=pre_post_Visits_sitevars_freq;
run;

PROC EXPORT DATA= WORK.pre_post_Visits_sitevars_freq
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Count of Changes by Varname and Site.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="Differences";
RUN;
```

```
* List the pre/post values for the sites and varnames of interest *;

proc sort data=pre_post_Visits;
  by Site SubjectID varname;
run;

PROC EXPORT DATA= WORK.pre_post_Visits (where=(Site='Site02' and Varname in ('DiaBP','SysBP')))
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Site02_DiaBP_SysBP.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="Differences";
RUN;

* In addition to the ID variable combinations, produced datasets where the
* combination was in the BASE dataset only or the COMPARE dataset only;

proc sql;
create table visits_base_only (drop=key) as
  select *, cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
    from CompareOut_Visits
    where _TYPE_='BASE' and
          cat(trim(Site),trim(SubjectID),trim(VisitType)) not in
      (select cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
         from CompareOut_Visits
         where _type_='DIF')
    order by key;
create table visits_compare_only (drop=key) as
  select *, cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
     from CompareOut_Visits
     where _TYPE_='COMPARE' and
          cat(trim(Site),trim(SubjectID),trim(VisitType)) not in
       (select cat(trim(Site),trim(SubjectID),trim(VisitType)) as key
          from CompareOut_Visits
          where _type_='DIF')
  order by key;
quit;


PROC EXPORT DATA= WORK.visits_base_only
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Rows_only_in_the_BASE_Dataset.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="BASE only Rows";
RUN;


PROC EXPORT DATA= WORK.visits_compare_only
            OUTFILE= "H:\MWSUG Fall 2018\Reports\Rows_only_in_the_COMPARE_Dataset.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="COMPARE only Rows";
RUN;
```