PharmaSUG 2022 - Paper AD-115

Checking Compliance of CDISC SDTM Datasets Utilizing SAS Procedures

Ballari Sen, Bristol Myers Squibb

ABSTRACT

CDISC (Clinical Data Interchange Standards Consortium) SDTM (Study Data Tabulation Model) is a required standard data structure to be used for submitting tabulation data to the US FDA and PMDA and NMPA submissions for clinical trials. An approach for implementing such data standards should be quick and effective with less resource utilization. Compliance Measures are needed to check compliance and streamline operations for the preparation of data for FDA, PMDA and NMPA submission, although there are many good validation compliance tools for CDISC SDTM ready compliance data sets such as Pinnacle 21 Enterprise, SAS - Clinical Standards Toolkit and OpenCDISC Validator [1][2]. However, the reusability, efficiency and automation of SAS® provides an effective and customized tool for validating CDISC SDTM compliance data sets to be used in clinical FDA submissions.

This paper will discuss how to efficiently produce the validation reports utilizing SAS® software utilizing base SAS capabilities. Validation is an essential part of the CDISC SDTM data sets, summary table and listing development process for elaborating the diverse approaches SAS programs can be applied to catch errors in data standard compliance or to identify inconsistencies that would otherwise be missed by any other general tool utilities. Catching errors in SDTM during validation prior to a submission can mean the difference between success or failure for a drug or medical device [2].

This paper also addresses the unclean raw data issues which could mess up the programming logic and warning messages and could ultimately lead to wrong or inaccurate conclusions and results impacting the drug's safety and efficacy, which could in turn be put the patient's health in jeopardy and have a significant negative impact on the pharmaceutical company [8].

INTRODUCTION

The FDA has endorsed CDISC SDTM as the preferred model for submitting clinical and bioequivalence data in the eCTD guidance [3]. Pharmaceutical organizations have adopted the SDTM model both because of FDA submission requirements [5] and because of benefits to the industry in communicating and sharing standardized data [3]. In clinical trials SDTM data sets development is a critical step in transforming non CDISC Complaint data sets coming from databases into a standard format for FDA submission and building the foundation for downstream analysis and reporting (ADaM, TLFs, Define.xml etc.) [4]. As the clinical industry becomes increasingly involved in developing CDISC Compliant SDTM data sets more effectively and efficiently it is mandatory to develop and execute SAS Validation programs to check compliance and streamline operations for submission ready files accordingly to SDTMIG (Study Data Tabulation Model Implementation Guide) [3].

The motivating factor for developing the validation programs was the requirement to generate and validate over 30 domains within a short period of time with limited resources. All the outputs were programmed using SAS® programs and utilizing SAS Macro programming for efficient outcome. Manually comparing the SDTM output to the validation output would have not been feasible in the allotted time without additional resources, also would have been prone to human error. In the event of accessing SDTM domain transformation, we realized it is at a high risk of potential errors and a duplicate programming was applied. This means an independent SAS programmer develops a separate program to read the same source EDC (Electronic Data Capture) data to generate the SDTM domains.

DESCRIPTION

The validation program will utilize the same mapping specifications to create the validation SDTM domains, however it will not use or share any of the same code logic which is used to derive the SDTM domains for submission. For example, if the developer generates a program through tool-based software for domain ae (Adverse Event), in order to generate the same validation adverse event domain (ae.sas7bdat) a separate and independent SAS Validation program is created and coded by another SAS

Clinical SAS Data Programmer. It will then generate another data set named val_ae.sas7bdat. The validation SAS Programmer then performs a proc compare between the AE Domain and the val_ae SAS data sets to identify the differences. proc compare method will be used to validate that a data set has been created correctly or that the changes in a data set are only those that are expected, to further notice if the observations were added or removed or corrections were done. We can also go ahead and distinguish and spot the difference between the variables in both the data sets with respect to: Type, length, formats, informats and label(s).

PROCEDURES

The following steps are performed to accomplish the duplicate programming validation process:

Program Header

Program headers are created to provide a high-level information about the program. Also, it includes traceability about the program authors and reference to programs.

```
/*--------------------------------------------------------------/

/ Program:        m-sdtm-Val       <Program Root Name>

/ Programmer:   Ballari Sen        <Author –>

/ Date:             20220301         <Original Creation Date – use yyyymmdd>

/ Project/Study: xxx-001            <Project or Study>

/ Purpose:  To Check Compliance of CDISC SDTM Data sets Utilizing SAS Procedures

/ Modification History:

/ Date         Version    Programmer     Description

/ -------- ------- ---------- ---------------------------------

/ 20220106  V01       Ballari Sen          initial version, copied from

                                                        /xxx/xxx/xxx/data/xx/001/xxx/xxxx/xxxx/xxx/xxx/

/- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -/

/ lib=   = Library information from where the data sets are getting copied.

/- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -/

/ Program Description:
```

- To create an independent SAS program to generate a validation AE data set.
- Compare the SDTM AE data set with the validation AE data set:
    - ➢ To check for perfect match output.
    - ➢ To check for data set summary for mismatched output.
    - ➢ To check for variable summary for mismatched output.
    - ➢ To check for observation summary for mismatched output.
    - ➢ To check for value comparison results for mismatched output.
    - ➢ To check for value discrepancies in variables generated by out, outbase, outcomp, outdif and outpercent options.
    - ➢ To check for variable attribute and meta-data mis-match output between sdtm ae data set and validation ae data set.

- To check for variables which are present, dropped and added in validation data set but not in sdtm data set and generate a separate data set to create a potential PDF report.

/---*/

Step 1 : Independent Clinical SAS Programmer and Folder Naming Convention:

The role of the data project manager of the SDTM projects is to assign an independent SAS Programmer to perform validation, and another programmer to generate SDTM data set. The validation SAS Programmer will develop a program VAL_AE SAS data set in a separate folder as compared to the SDTM Developer programmer. She/he is given the same information such as the Protocol, annotated CRF and SDTM Mapping specifications. The validation Programmer will generate a similar data set VAL_AE as the SDTM AE Data set.

Step 2 : Creating library name for the clinical trial study datasets:

The validation SAS Programmer needs to create & define a different set of libnames for viewing the Raw Data sets & the SDTM Data sets created by the SDTM Developer.

```
dm 'log; clear; output; clear; odsresults; clear;

%let prognam = Drive:\Clinical Trial Data sets\xxxxxx\XXXX-X-
XXX\SAS\QC\Results\QC\Output_Log ;

proc printto log = "&prognam\ae_log.log" new;

proc datasets lib=work kill nolist;
run;
quit;

/*****1. Locations of data and files (RAW, SDTM, ADaM, excel)****/
/*****2. Locations of Utility programs/macros/formats etc.******/

libname raw "Drive:\Clinical Trial Datasets\XXXXX\XXXX-X-XXX\Export\raw
Datasets " access=readonly ;

libname sdtm "Drive:\Clinical Trial Datasets\XXXXX\XXXX-X-XXX\Export
\Deliverables\sdtm" access=readonly;
```

Step 3 : Reading the Mapping Specification:

The SDTM Specifications are stored in a standard EXCEL Mapping specification format. The Validation programmer must generate the validation SDTM datasets from the information stored in the specifications.

```
/******Import the SDTM Mapping Specification Excel sheet***********/

%let specpath1 = %str(R:\Clinical Trial Datasets\Studies\XXXX\XXXXX-X-
XXX\SDTM Mapping Specification\XXXX-X-XXX SDTM Mapping Specification);

proc import out=work.ae_map
datafile= "&specpath1"
dbms=xlsx replace;
   sheet='Variable';
run;
```

Step 4 : Defining the Domain Template & Generating the Validation Domain

The actual programming coding logic which is used to transform the raw data sets to SDTM data sets can be elaborate and unique to each programmer and SDTM Domain.

The below example illustrates how the SAS programmer utilizes DATA step & PROC SQL Procedures to manipulate data sets and macro variables. The Developer might have utilized some other SAS Procedures or maybe have utilized an in-house tool to generate the Target/ Production AE SDTM Data set.

Once the duplicate validation programmed data set is created, further it is sorted and compared with the target SDTM data set. The main motive for this step is for the validation programmer to research and analyze the differences in order to understand and determine if it is an error from the validation program or in the original/Target SDTM data set.

Once the issue has been identified, the validation programmer must clearly report the data mismatch findings to the original SDTM developer programmer and the project manager to have the error resolved. This entire process does ensure for more accurate results and catches data discrepancies.

```
    /***Sorting the raw ae and raw.sae data sets *********/

        proc sort data=raw.ae out=r_ae1;
      by subject recordspid;
       run;

      proc sort data=raw.sae out=r_sae1 (rename=(aespid = recordid)
       drop= recordid);
         by subject aespid;
             run;

  /*Merging sorted raw.ae and raw.sae datsetts ;
     Excluding Records where the AETERM is Null */

           data temp_ae_;
             merge r_ae1 r_sae1;
           by subject recordid;
         if /*AEYN='Y' and*/ not missing(AETERM);
       rename aescong_1=_aescong aesdisab_1=_aesdisab aesdth_1=_aesdth
  aeshosp_1=_aeshosp aeslife_1=_aeslife aesmie_1 =_aesmie;
      drop studyid;
       run;

   /*** Define the code list used to generate standard sdtm ***/

  proc format;
      value YN_SDTM
            0 = 'No'
            1 = 'Yes'
            3 = 'Unknown'
            ;
         value $Mon_SDTM
            'JAN' = '01'
            'FEB' = '02'
            'MAR' = '03'
            'APR' = '04'
            'MAY' = '05'
            'JUN' = '06'
            'JUL' = '07'
            'AUG' = '08'
            'SEP' = '09'
            'OCT' = '10'
            'NOV' = '11'
            'DEC' = '12'
             other = ' '
            ;
       value $YN_SDTM1
            'Yes' = 'Y'
            'No'  = 'N'
            ;
           run ;

 /***Data mapping & derivation***;
 ***   1. Source data used***;
 ***   2. Variable mapping/derivation***;/

       data temp_ae_F;

       length aeacn $50;

       set temp_ae_;

 * 1.Common variables in all data sets;
      USUBJID = "XXXXX-X-XXX" || "-" || strip(subject);

 * 2.Manual assigned varialbes;
      STUDYID = "XXXX-X-XXX";
      DOMAIN = "AE";

 * 3.Rename or mapping variables only for AE domain;
      AESPID = strip(put(recordid1,best.));

      AETERM  = strip(upcase(aeterm));
      AELLTCD = input(aeterm_llt_code_1, best.);
      AEPTCD  = input(aeterm_pt_code_1,  best.);
      AEHLTCD = input(aeterm_hlt_code_1, best.);
      AEHLGTCD= input(aeterm_hlgt_code_1,best.);
      AEBDSYCD= input(aeterm_soc_code_1, best.);
      AESOCCD = input(aeterm_soc_code_1, best.);
      AEBODSYS = aesoc;
      AESOC =    aesoc;
       AESER =    substr(AESER,1,1);

 /**De-coding variables according to SDTM Format***/

   if aeacn1dn = 1 then AEACN = 'DOSE NOT CHANGED';
   else if aeacn1dr = 1 then AEACN = 'DOSE REDUCED';
   else if aeacn1dt = 1 then AEACN = 'DRUG INTERRUPTED';
   else if aeacn1dw = 1 then AEACN = 'DRUG WITHDRAWN';
   else if aeacn1na = 1 then AEACN = 'NOT APPLICABLE';
   else if aeacn1un = 1 then AEACN = 'UNKNOWN';

            aerel=upcase(aerel1);
            aeout=upcase(aeout);

    if   _aescong =1 then AESCONG ='Y';
 if _aesdisab=1 then AESDISAB='Y';
 if aecmyn_RAW ='No Treatment Given' then aecontrt='N';
 if aecmyn_RAW in ('Procedure Performed','Drug Therapy Administered','Drug
 Therapy Administered & Procedure Performed','Other') then aecontrt ='Y';

 /***Populating date variables according to ISO8601 SDTM format****/

 if not missing(stdat) then aestdtc = put(stdat,is8601dn.);
 else if missing(stdat) and not missing(stdat_raw) then do;
 year=scan(stdat_raw,3);
 month=put(upcase(strip(scan(stdat_raw,2))),$mon.);
 if not missing(year) and missing(month) then aestdtc = year;
 else if not missing(year) and not missing(month) then aestdtc = strip(year)
 ||'-' || strip(month);
      end;

 if not missing(endat) then aeendtc = put(endat,is8601dn.);
 else if missing(endat) and not missing(endat_raw) then do;
 year=scan(endat_raw,3);
 month=put(upcase(strip(scan(endat_raw,2))),$mon.);
 if not missing(year) and missing(month) then aeendtc = year;
 else if not missing(year) and not missing(month) then aeendtc = strip(year)
 || '-' || strip(month);
      end;
 run;

 /**Populating Sequence Variable : AESEQ *************/
 proc sort data=temp_ae1;
      by usubjid aedecod aestdtc aespid ;
 run;

 data temp_ae1;
      set temp_ae1;
      by usubjid aedecod aestdtc aespid ;
      if first.usubjid then AESEQ = 1;
      else aeseq +1;
 run;

 /**** Merge with sdtm.dm to get Reference state date***********/
 data dm;
 set sdtm.dm;
 keep usubjid rfstdtc rfendtc;
 run;
 proc sort data=dm;
 by usubjid; run;

 proc sort data=temp_ae_f;
 by usubjid; run;

 /*****Deriving aestdy, aeendy and epoch Variables********/
 data temp_ae_f1;
 merge temp_ae_f(in=in1) dm(in=in2);
 by usubjid;
 if in1;
 if length(aestdtc)=10 and not missing(rfstdtc)
 then aestdy = input(aestdtc, is8601da.) - input(rfstdtc, is8601da.) +
 (input(aestdtc, is8601da.) >= input(rfstdtc,is8601da.));
```

```
if length(aeendtc)=10 and not missing(rfstdtc) then aeendy = input(aeendtc,
is8601da.) - input(rfstdtc, is8601da.) + (input(aeendtc, is8601da.) >=
input(rfstdtc, is8601da.));
if aestdtc = '' then epoch = '';
else if aestdtc < rfstdtc or rfstdtc = '' then epoch = 'SCREENING';
else if aestdtc <= rfendtc then epoch = 'TREATMENT';
else epoch = 'FOLLOW-UP';
run;

/**combining all variables into one parent ae data set***************/

proc sql noprint ;
create table ae_qc(label="adverse event") as
select studyid  label="study identifier",
domain  label="domain abbreviation",
usubjid label="unique subject identifier",
aeseq   label="sequence number",
aespid  label="sponsor-defined identifier",
aeterm  label="reported term for the adverse event",
aellt   label="lowest level term",
aelltcd label="lowest level term code",
aedecod label="dictionary-derived term",
aeptcd  label="preferred term code",
aehlt   label="high level term",
aehltcd label="high level term code",
aehlgt  label="high level group term",
aehlgtcd label="high level group term code",
aebodsys label="body system or organ class",
aebdsycd label="body system or organ class code",
aesoc    label="primary system organ class",
aesoccd  label="primary system organ class code",
aeser    label="serious event",
aeacn    label="action taken with study treatment",
aeacnoth label="other action taken",
aerel    label="causality",
aeout    label="outcome of adverse event",
aescong  label="concomitant or additional trtmnt given",
aesdisab label="persist or signif disability/incapacity",
aesdth   label="results in death",
aeshosp  label="requires or prolongs hospitalization",
aeslife  label="is life threatening",
aesmie   label="other medically important serious event",
aecontrt label="congenital anomaly or birth defect",
epoch    label="epoch",
aestdtc  label="start date/time of adverse event",
aeendtc  label="end date/time of adverse event",
aestdy   label="study day of start of adverse event",
aeendy   label="study day of end of adverse event",
aeenrtpt label="end relative to reference time point",
aeentpt  label="end reference time point"
from temp_ae_f1
where not missing(aeterm);
quit;
```

```
          /******SUPPAE QNAM Derivation ************/

  /***creation of internal suppqual macro for raw ae variables ***/

%macro create1(qnam=,qlabel=,var1=);
         select "XXXXX-X-XXX" as STUDYID,
                  'AE' as RDOMAIN,
                  USUBJID,
                  'AESEQ' as IDVAR,
                  compress(put(aeseq, 12.)) as IDVARVAL,
                  &qnam as QNAM length = 8,
                  &qlabel as QLABEL length = 40 ,
                  &Var1 as QVAL length = 200 ,
                  "CRF" as QORIG,
                  "" as QEVAL
         from temp_ae_f1
         where &var1 is not missing
%mend;


%macro create2(qnam=,qlabel=,var1=);
         select "XXXXX-X-XXX" as STUDYID,
                  'AE' as RDOMAIN,
                  USUBJID,
                  'AESEQ' as IDVAR,
                  compress(put(aeseq, 12.)) as IDVARVAL,
                  &qnam as QNAM length = 8,
                  &qlabel as QLABEL length = 40 ,
                  Case
                  When &Var1 = "YES" then "Y",
                  When &Var1 = "NO" then "N",
                  else ""
                  end as QVAL length = 200 ,
                  "CRF" as QORIG,
                  "" as QEVAL
         from temp_ae_f1
         where &var1 is not missing
%mend;

/***Populating the table final supp_base to contain all the suppae
values**/

proc sql noprint;
   create table supp_base as

/* QNAM = "suppqual1" */
     Select "xxxxxx" as studyid,
     "AE" as RDOMAIN,
      USUBJID,
      "AESEQ" as IDVAR,
      Compress(put(aeseq,12.)) as IDVARVAL,
      "AEXXXX" as QNAM length = 8 ,
      "AEXXXX" as QLABEL length = 40 ,
       Var1 as QVAL length = 200 ,
       "CRF" as QORIG,
        "" as QEVAL
        from base
```

```
    union all

    /* QNAM = "suppqual2" */

/* QNAM = "AEXXXXX"; qlabel=%str('suppqual2_label')* /
    %create1(var1=AEXXXXX)
    order by usubjid;

    union all

    /* QNAM = "suppqual2" */

/* QNAM = "AEXXXXX"; qlabel=%str('suppqual3_label')* /
    %create2(var1=AEXXXXX)
    order by usubjid;


    union all

    /* QNAM = "suppqual4" */

/* QNAM = "AEvaluex" ; qlabel=%str('suppqual4_label')* /
    %create3(var=AEvaluex )
    order by usubjid;
quit;

/******Generating the table for Suppqual AE Domain*********/

proc sql noprint;
create table suppae_v as
select studyid label='study identifier' length=20,
rdomain label='related domain abbreviation',
usubjid label='unique subject identifier' length=29,
idvar label='identifying variable' length=8,
idvarval label='identifying variable value' length=40,
qnam label='qualifier variable name',
qlabel label='qualifier variable label',
qval label='data value',
qorig label='origin' length=40,
qeval label='evaluator' length=40
from supp_base
where qval not in('')
order by usubjid, idvarval, qnam;
quit;
```

Step 5: Comparison of the SDTM.AE & Validation.AE datasets utilizing proc compare SAS procedure.

proc compare is a powerful procedure in SAS® that allows users to compare two data sets, to compare variables against variables of the same data set or variables against variables between two data sets [7].

```
/**** Data comparison: sdtm.ae vs ae(validation)
using proc compare sas procedure*********/

proc sort data=ae_qc out = ae_v ;
    by usubjid aespid aedecod aestdtc ;
```

```
run;

proc sort data=sdtm.ae out=ae_sdtm ;
   by usubjid aespid aedecod aestdtc ;
run;

proc compare data=ae_sdtm compare=ae_v out=check listall listvar listobs
listbase outbase outcomp outdif outnoeq;
id usubjid aebodsys aedecod aeterm aestdtc aeendtc;
run;


/**** Data comparison: sdtm.suppae vs suppae (validation) using proc
compare sas procedure **********/

proc sort data=sdtm.suppae out=suppae_sdtm ;
by usubjid qnam qlabel qval;
run;

proc sort data= suppae_V;
by usubjid qnam qlabel qval;
run;

proc compare data= suppae_sdtm compare=suppae_V out=check listall listvar
listobs listbase outbase outcomp outdif outnoeq;
run;
```

COMPARING TWO SAS® LIBRARIES

In the clinical SAS industry comparing and validating data sets by parallel programming and comparing data sets is a common task, but due to constant updates and changes in raw data attributes and variable changes it becomes increasingly difficult to figure out the differences between two data sets. This paper also describes to demonstrate PROC COMPARE'S and SASHELP directories capability to identify the data sets present in various SAS® libraries by listing the mismatched data sets. Secondly to list out the observation and variable differences in both the data sets by listing out the total number of observations and variables present in both the data sets and also identifying the variable mismatch and the differences in observation number [8].

PROCEDURES

Data Extraction

The data from the libraries are extracted with the help of the macro %check. The macro creates two data sets per library: sdtmobs and valobs and sdtmvar and valvar. libobs contains all the data sets along with the number of observations as present in the library which are extracted from sashelp.vtable. The libvar data set contains detailed information on the data set's name, column name, length, label, format, informat and type that are extracted from sashelp.vcolumn [8].

```
%macro check (library);
proc sql;
create table &library.obs as
select distinct memname, nobs as &library.obs
from sashelp.vtable
where libname eq "&library." and memtype eq "data"
order by memname;

create table &library.var as
```

```
select libname as &library._libname, memname as &library._memname, name,
length as &library._length,
label as &library._label, format as &library._format, informat as
&library._informat, xtype as &library._xtype
from sashelp.vcolumn
where libname eq "&library." and memtype eq "data"
order by memname, name;
 quit;
%mend check ;
%check(val);
%check(sdtm);
```

Observation Differences between two datasets in two different libraries

The sdtmobs and valobs data sets are used to find two major differences between both libraries by merging them into a data set and populating a flag variable which flags the missing data sets as "Missing : sdtm/val data set". If the data sets are populated in both the libraries, the no. of observations are compared and the observation difference are assigned to the flag variable as "Difference : No of observations (N)" where N represents the number of observations that differs in both the data sets. Another flag variable is created is which describes the number of observation in each of the sdtmobs and valobs data sets [8].

```
data observtns ;
merge sdtmobs (in = a) valobs (in = b);
by memname;
format flag_diff $90.;
if sdtmobs ^= valobs then flag_diff = propcase("Difference : No. of
observations(")|| compress(put(abs(sdtmobs - valobs),8.))||")";
if valobs = . then flag_diff = propcase("Not Present : Validation dataset")
;
if sdtmobs = . then flag_diff = propcase("Not Present : SDTM dataset") ;
if flag_diff ne "";
if scan(flag_diff,1) = propcase("Difference")
then variables = propcase("valobs = ") || compress(put(valobs,8.)) ||
propcase(" :  sdtmobs = ") || compress(put(sdtmobs,8.)) ;
```

Variable Property Differences between two datasets in two different libraries

The sdtmvar and valvar data sets are used to find the differences in the metadata of the variables which include label, length, format, informat and type. Sdtmvar and valvar data sets are merged by name variable to contain the variable attributes of both the libraries. The difference between both the data sets are then flagged to the difference_flag variable to populate as "Y" whenever there is a difference between the variable's meta-data attributes. Val_qc data set populated contains variables which are present in the validation data sets but not populated in the sdtm data set. Sdtm_qc data set populated contains variables which are present in the sdtm data sets but not populated in the validation data set.Val_Sdtm_Meta_Diff data set populated contain the variable where there is a difference between their variable attributes (label,length,format,informat and type) between the validation and sdtm data set.

```
/**Compare variable attributes differences between Validation & SDTM
datasets***/

 data val_qc sdtm_qc val_sdtm_Meta_Diff ;
 merge valvar (in=a) sdtmvar (in=b);
 by name;
```

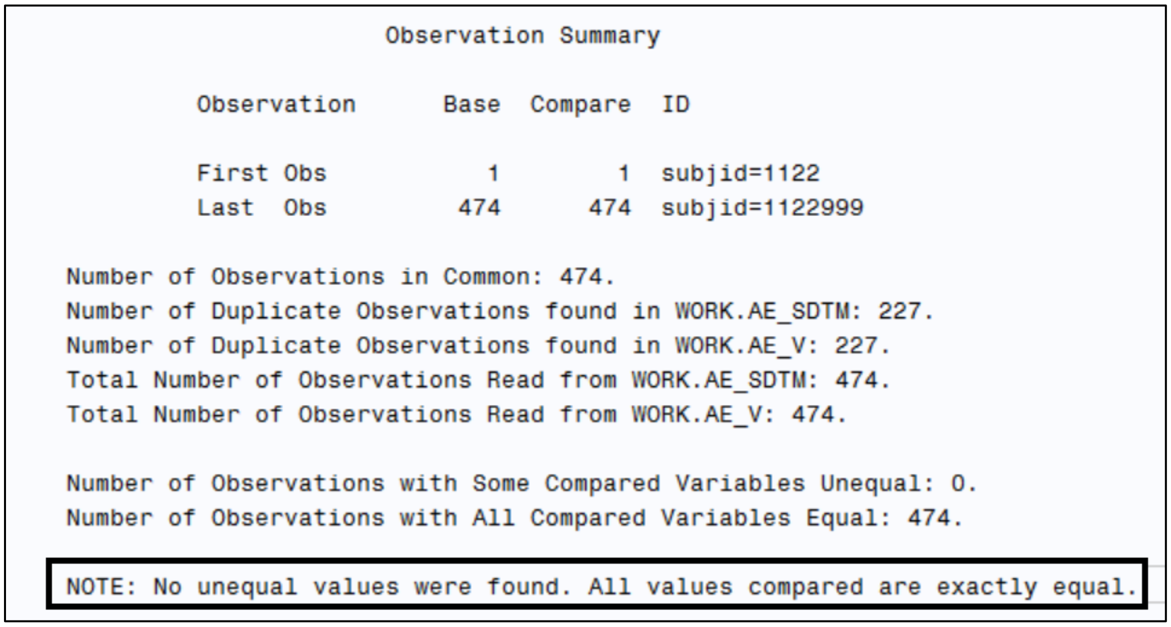
```
if a and not b then output val_qc ;
else if b and not a then output sdtm_qc;
else do;
if VAL_Xtype ^= SDTM_Xtype then diff_flag ="Y";
if VAL_Length ^=SDTM_Length then diff_flag ="Y";
if VAL_Label ^=SDTM_Label then diff_flag ="Y";
if VAL_Format ^=SDTM_Format then diff_flag  ="Y";
if VAL_Informat ^=SDTM_Informat then diff_flag  ="Y";
if flag="Y" then output Val_Sdtm_Meta_Diff;
end;
run;
```

PDF report can also be generated to populate the results as the final output.

```
ODS PDF File="Difference.pdf";
title1 "Variable Difference";
proc report data= Val_Sdtm_Meta_Diff ;
columns variable val_label sdtm_label ;
define variable / display;
define val_label / display;
define sdtm_label / display;
run;
title1;
ods pdf close;
```

RESULTS

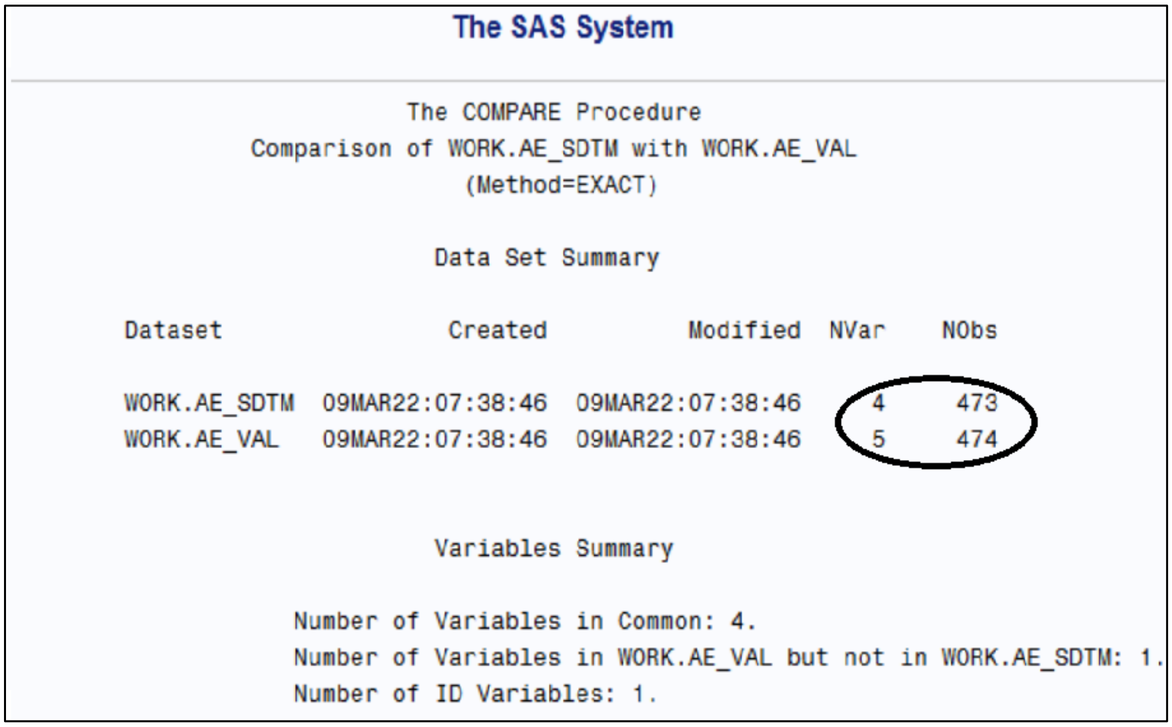
The resulting proc compare output, where there is a perfect match between the datasets is shown in Figure 1:



```
                       Observation Summary

           Observation      Base  Compare  ID

           First Obs           1        1  subjid=1122
           Last  Obs         474      474  subjid=1122999

   Number of Observations in Common: 474.
   Number of Duplicate Observations found in WORK.AE_SDTM: 227.
   Number of Duplicate Observations found in WORK.AE_V: 227.
   Total Number of Observations Read from WORK.AE_SDTM: 474.
   Total Number of Observations Read from WORK.AE_V: 474.

   Number of Observations with Some Compared Variables Unequal: 0.
   Number of Observations with All Compared Variables Equal: 474.

   NOTE: No unequal values were found. All values compared are exactly equal.
```

Figure 1: Perfect Match Output

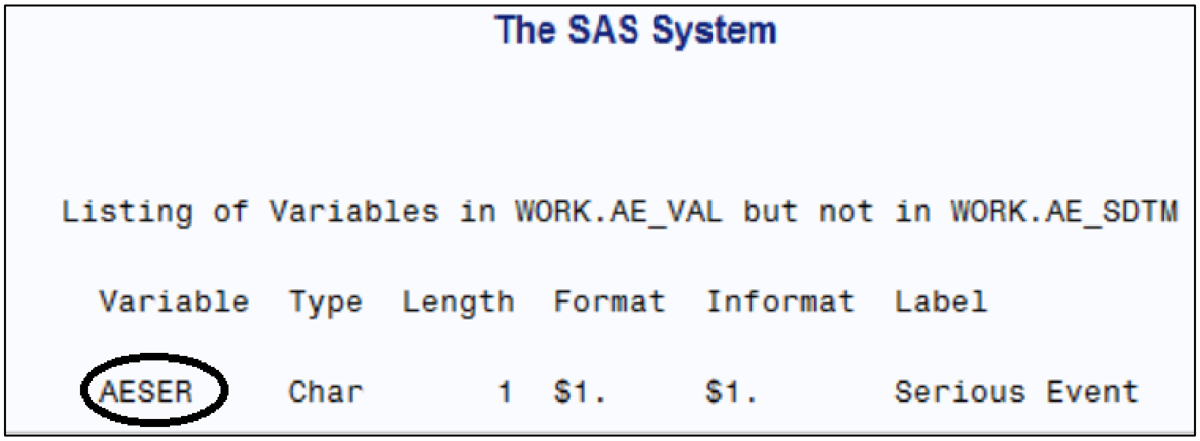
The resulting proc compare output, where there is not a match between the datasets is shown in Figure 2: Based on the Data set summary report there are discrepancies in the number of variables and the number of observations in the 2 datasets.



```
                         The SAS System

                     The COMPARE Procedure
           Comparison of WORK.AE_SDTM with WORK.AE_VAL
                        (Method=EXACT)

                       Data Set Summary

   Dataset              Created          Modified  NVar    NObs

   WORK.AE_SDTM  09MAR22:07:38:46  09MAR22:07:38:46     4     473
   WORK.AE_VAL   09MAR22:07:38:46  09MAR22:07:38:46     5     474


                       Variables Summary

           Number of Variables in Common: 4.
           Number of Variables in WORK.AE_VAL but not in WORK.AE_SDTM: 1.
           Number of ID Variables: 1.
```

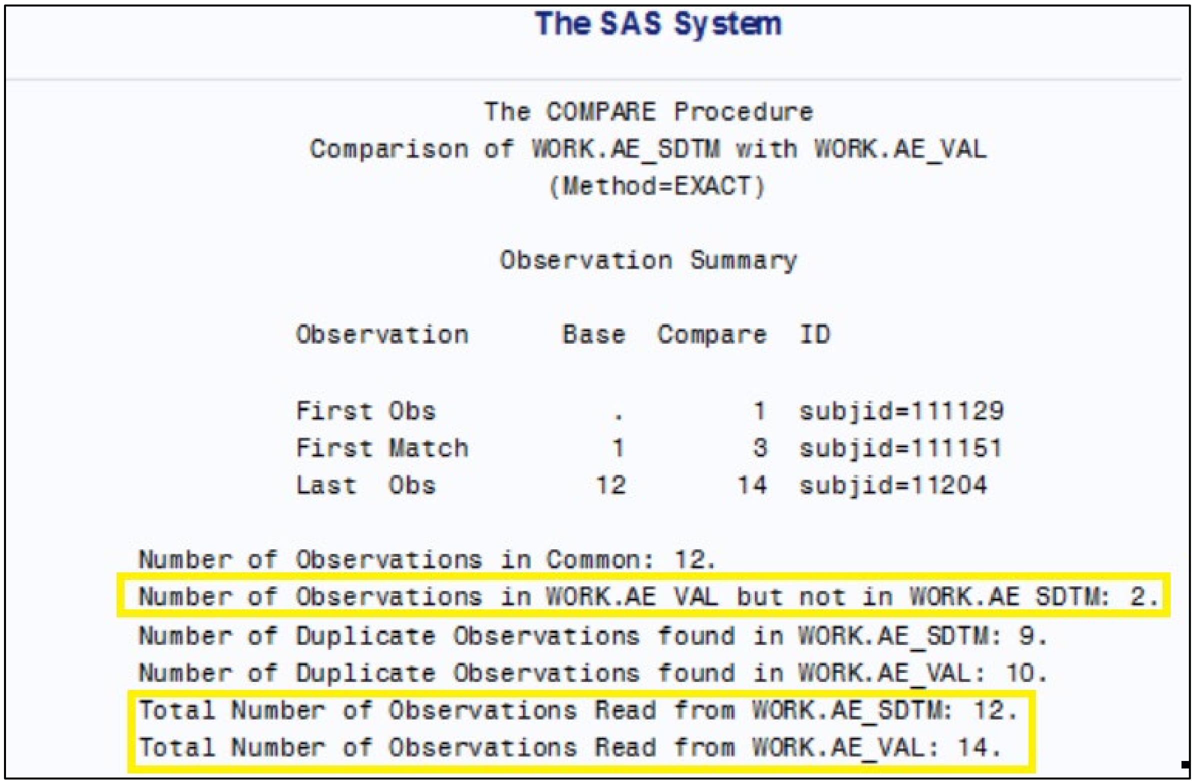
Figure 2: Dataset Summary for Mismatched Output

Glancing over the Variable Summary Report below, the result is as expected. Just by observing the datasets, one can see that the variable AESER exists only in validation.AE dataset.



```
                         The SAS System



 Listing of Variables in WORK.AE_VAL but not in WORK.AE_SDTM

  Variable  Type  Length  Format  Informat  Label

  AESER     Char       1  $1.     $1.       Serious Event
```

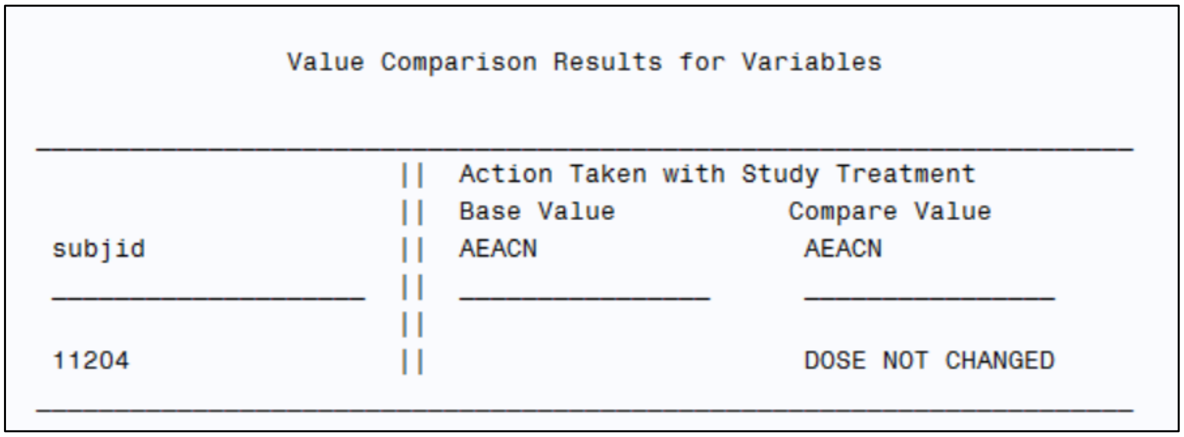
Figure 3: Variable Summary for Mismatched Output



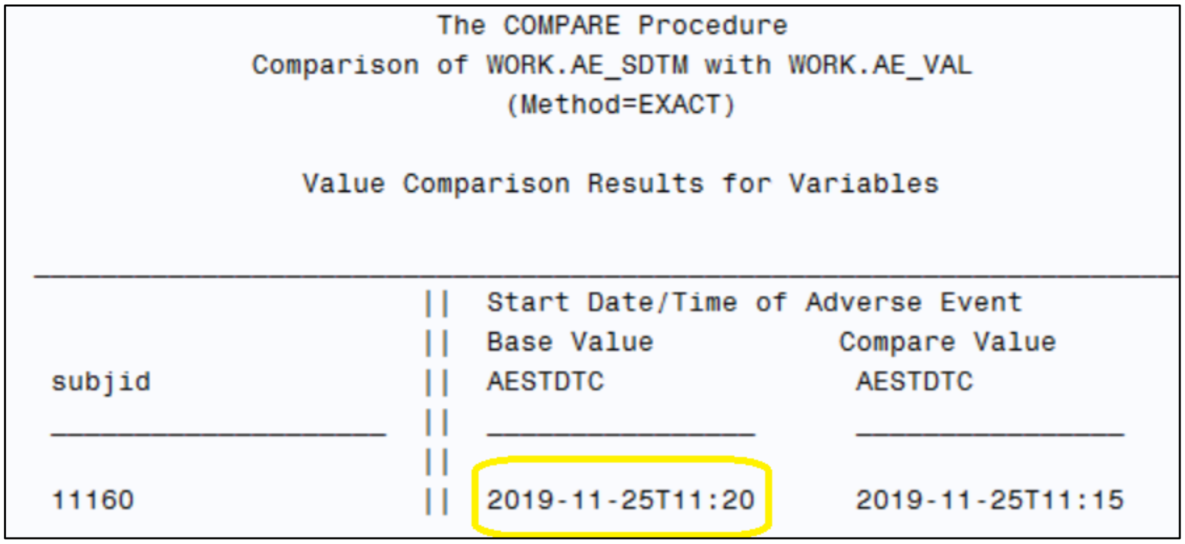
```
The SAS System

                    The COMPARE Procedure
         Comparison of WORK.AE_SDTM with WORK.AE_VAL
                       (Method=EXACT)

                    Observation Summary

          Observation      Base  Compare  ID

          First Obs           .        1  subjid=111129
          First Match         1        3  subjid=111151
          Last  Obs          12       14  subjid=11204

Number of Observations in Common: 12.
Number of Observations in WORK.AE VAL but not in WORK.AE SDTM: 2.
Number of Duplicate Observations found in WORK.AE_SDTM: 9.
Number of Duplicate Observations found in WORK.AE_VAL: 10.
Total Number of Observations Read from WORK.AE_SDTM: 12.
Total Number of Observations Read from WORK.AE_VAL: 14.
```

Figure 4: Observation Summary for Mismatched Output

The observation summary report shows the discrepancy as there are two observations that are present in the qc.ae but not present in prod sdtm.ae.



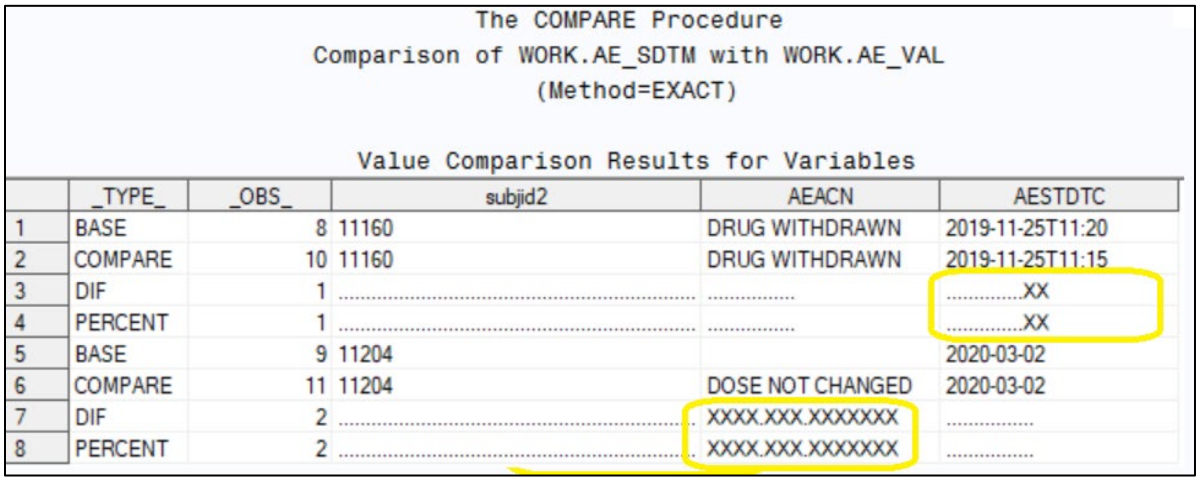
```
                 Value Comparison Results for Variables

__________________________________________________________________
                      ||  Action Taken with Study Treatment
                      ||  Base Value           Compare Value
subjid                ||  AEACN                 AEACN
____________________  ||  _______________       _______________
                      ||
11204                 ||                        DOSE NOT CHANGED
__________________________________________________________________
```



```
                       The COMPARE Procedure
            Comparison of WORK.AE_SDTM with WORK.AE_VAL
                          (Method=EXACT)

                Value Comparison Results for Variables

____________________________________________________________________
                          ||  Start Date/Time of Adverse Event
                          ||  Base Value          Compare Value
  subjid                  ||  AESTDTC              AESTDTC
  _____________________   ||  ________________     ________________
                          ||
  11160                   ||  2019-11-25T11:20     2019-11-25T11:15
```

Figure 5 : Value Comparison Results for Mismatched Output

The value comparison results for variable report displays the discrepancies in variables. Here, one sees discrepancies in aeacn and aestdtc variables. Note that the value for aeacn is not present for subjid = "11160" in the prod sdtm.ae dataset. Also the value of aestdtc = "2019-11-25T11:20" is not matching with qc.ae dataset.



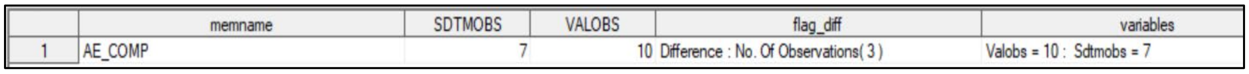
```
                   The COMPARE Procedure
        Comparison of WORK.AE_SDTM with WORK.AE_VAL
                      (Method=EXACT)

          Value Comparison Results for Variables
```

	TYPE	_OBS_	subjid2	AEACN	AESTDTC
1	BASE	8	11160	DRUG WITHDRAWN	2019-11-25T11:20
2	COMPARE	10	11160	DRUG WITHDRAWN	2019-11-25T11:15
3	DIF	1	....	....	XX
4	PERCENT	1	....	....	XX
5	BASE	9	11204		2020-03-02
6	COMPARE	11	11204	DOSE NOT CHANGED	2020-03-02
7	DIF	2	....	XXXX.XXX.XXXXXXX	....
8	PERCENT	2	....	XXXX.XXX.XXXXXXX	....

Figure 6 : Out_ae1 mismatched datasets generated by out = out_ae1,OUTBASE, OUTCOMP,OUTDIF and OUTPERCENT options .

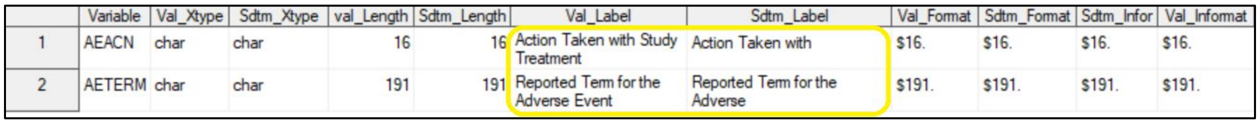
The use of OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT and OUT= options in the above code enables one to create an output data set with values from BASE and COMPARE data sets, the difference and the percent difference. OUTNOEQUAL option instructs the COMPARE procedure to only output observations with discrepancies [7]. The use of NOPRINT suppresses the printing options. But the mention of an output data set is essential otherwise sas will ignore the OUTBASE, OUTCOMP, OUTDIF, OUTPERCENT and OUTNOEQUAL options.

Out_ae1 contains only observations that are not a match. When the value of the ID variable exists in both the base and compare data sets (subjid = "11160") the program creates two observations of _TYPE_ = "BASE", "COMPARE" ,"DIFF",""PERCENT" . _TYPE_ = "BASE" & "COMPARE" represents data sets values from Base & Compare data sets. _TYPE_ = "DIFF" represents the difference and _TYPE_ = "PERCENT" shows the percent difference.



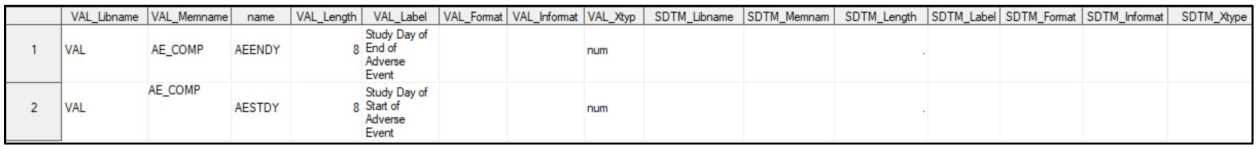
	memname	SDTMOBS	VALOBS	flag_diff	variables
1	AE_COMP	7	10	Difference : No. Of Observations(3)	Valobs = 10 : Sdtmobs = 7

Figure 7 : Observation mismatch output between SDTM data set and Validation dataset . The difference in observations were 3. The validation dataset had 10 observations and the SDTM datasets had 7 observations [8].



	Variable	Val_Xtype	Sdtm_Xtype	val_Length	Sdtm_Length	Val_Label	Sdtm_Label	Val_Format	Sdtm_Format	Sdtm_Infor	Val_Informat
1	AEACN	char	char	16	16	Action Taken with Study Treatment	Action Taken with	$16.	$16.	$16.	$16.
2	AETERM	char	char	191	191	Reported Term for the Adverse Event	Reported Term for the Adverse	$191.	$191.	$191.	$191.

Figure 8 : Variable attribute mis-match output between SDTM dataset and Validation dataset . For the variable AEACN the validation dataset label = "Action taken with Study Treatment" but the SDTM dataset label = "Action Taken with". The mismatch is populated at the Diff variable.



	VAL_Libname	VAL_Memname	name	VAL_Length	VAL_Label	VAL_Format	VAL_Informat	VAL_Xtyp	SDTM_Libname	SDTM_Memnam	SDTM_Length	SDTM_Label	SDTM_Format	SDTM_Informat	SDTM_Xtype
1	VAL	AE_COMP	AEENDY	8	Study Day of End of Adverse Event			num			.				
2	VAL	AE_COMP	AESTDY	8	Study Day of Start of Adverse Event			num			.				

Figure 9 : Variables mis-match output between SDTM dataset and Validation dataset . Variable AEENDY & AESTDY are present in the validation dataset but not present in the SDTM dataset.

CONCLUSION

The Validation programming that are described in this paper can be used as a vital programming code in reviewing clinical data. In addition, several additional data driven custom checks can be established based on the study design, protocol and CRF Design. A detailed validation of SDTM data sets is essential in a successful FDA submission. The CDISC SDTM Module is constantly being enhanced with more domains and variables and therefore adding to the growing complexity. SDTM mapping specification and its interpretation on how each SDTM variable will transform will vary for each organization [2].

Therefore, the need of a customized and flexible process is required to perform unique validation to each SDTM Transformation implementation. SAS being a statistical analysis software fulfills this criterion since the users get the liberty to perform custom programs & SAS Macro scripts to understand and transform the SDTM mapping specification into a duplicate validation domain. This paper provides tips and procedures to speed up the validation process. The purpose of the SAS validation programs is not to completely replace pinnacle 21 Enterprise but rather it enhances these standardized tools and increases its efficiency. SAS programs in combination with the existing validation tools can achieve higher levels of validation approach, further ensuring cleaner and more compliant SDTM data sets for downstream analysis and reporting [2].

REFERENCES

1. Kanevsky, Max. 2008."Validating SDTM, an open source solution". *Proceeding of the CDISC Interchange* 2008.
2. Truong, Sy.2017. "Automate Validation of CDISC SDTM with SAS®". *Paper 836-2017.*
3. Busa Bhavin and Vince Sheila and Aziz Jameelah. 2008. "Validating CDISC SDTM-Compliant Submission-Ready Clinical Datasets with an In-House SAS® Macro-Based Solution". *Paper RS07.*
4. Srivastva Abhinav. 2016. "Pre-Data Checks for SDTM Development*". MWSUG Paper PH01*.

5. Wood Fred and Guinter Tom. Evolution and Implementation of the CDISC Study Data Tabulation Model (SDTM). *Paper FC08.*
6. Larry Liu Lixiang.2017. “Statistician’s secret weapon: 20 ways of detecting raw data issues”. *PharmaSUG 2017 - Paper PO06.*
7. Cheng Alice M and Wise Michael R and Flavin Justina M. “How to Speed Up Your Validation Process Without Really Trying*”. Paper 10840-2016*
8. Janapala Bharat Kumar. “Finding All Differences in two SAS® libraries using Proc Compare”.

ACKNOWLEDGMENTS

I would like to thank my managers, Ankur Sharma and Todd Rider for their constant encouragement and valuable and constructive feedback. I would also like to thank Samar Noor for providing me the opportunity to present the paper at Pharma SUG Conference.

CONTACT INFORMATION

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

Name : Ballari Sen
Company : Bristol Myers Squibb
Email : Ballari.sen@bms.com