PharmaSUG 2013 - Paper PO01

LST in Comparison

Sanket Kale, Parexel International Inc., Durham, NC
Sajin Johnny, Parexel International Inc., Durham, NC

ABSTRACT

The need for producing error free programming deliverables combined with the increasing globalization of the pharmaceutical industry has led to the growing requirement for validation of programs associated with generating Clinical trial reports. Double programming is the most commonly accepted and widely used standard method in the industry for validation of programming algorithms and to crosscheck adherence to specifications. Double programming involves two programmers programming independently based on specifications and then utilizing procedures like COMPARE to match the outputs. Currently the accepted QC process followed in most companies requires programmers to manually review the Proc Compare results for verifying the similarity of the outputs produced. This manual process can be very cumbersome and resource intensive especially while handling large deliveries and also increases the likelihood of human error.

This paper looks at an innovative method of eliminating this manual process with a SAS macro that scans the proc compare output to check for similarity as well as listing any differences observed in the two outputs. The program will read in all .lst files in user specified folders in a sequential manner and search the contents of each file for existence of substrings to determine equivalence or list differences in the data compared. The derived results of all scanned .lst outputs will be summarized in one detailed report. The surmise is that this report will help improve the efficiency and quality of the validation process by eliminating the need for time consuming manual review, thereby eliminating the risk of human error while also providing sufficient documentation in support of audit perspective processes.

INTRODUCTION

The pharmaceutical industry is a well-regulated industry and quality control is of foremost importance. Amongst the several different methods of QC, the one which ensures a high level of quality is independently programming the dataset or report to be submitted. The final step in this process is to compare the independently produced outputs and ascertain that they match. The SAS procedure “Proc Compare” is used for the comparison of the independently programmed datasets. The Proc Compare typically produces an .LST file as its output. This .LST file contains a results report of the comparison of the 2 datasets. The .LST file is the equivalent of a .TXT file and can be opened by any of the text editing tools.

THE TIME-CONSUMING AND CUMBERSOME TRADITIONAL METHOD

To determine whether the independently created outputs are a match or not, this .LST file has to be checked. In many companies the method employed for checking this .LST file is often manual. The general steps for the manual process are outlined below:

- Open each compare output report.
- Check if the Time stamp for the QC dataset is after the production dataset to ensure that the Independent QC program has been run after the generation of the production dataset.
- Many companies also have a stringent requirement that at least one of the datasets used in the compare procedure should be from a controlled environment (primary, production, etc.). The other can be from SAS work environment. This check is to ensure that both the datasets are not from SAS work environment but are from the appropriate company defined environments.
- Check if the Labels are matching.
- Check if there are any additional variables in the production side which are not qc’ed or if there are any additional stray variables on the QC side which should be dropped. Many companies like to have the compare procedure output as clean as possible.
- Most times a perfect match is needed for sign-off. A perfect match for the sign-off on the output is to have all the above steps in order and all observations values, variables values and variable attributes between the two datasets should be exactly the same.
- After checking for all of the above, the programmers may have to print and sign the compare output for documentation purposes.

The above steps require a lot of manual process including opening each file, printing, checking and signing. This process, although essential, is cumbersome and time-consuming. A typical delivery of 300 outputs can take 10-12 hours just for following this process of printing, manual checking, signing and documentation of the Proc Compare outputs.

THE TIME-SAVING NEW APPROACH

The manual approach is cumbersome, prone to error and extremely time-consuming. The authors are proposing a new approach which automates the process entirely and can be used across studies or companies with equal ease. The highlights of the automated approach are:

- ✓ Checks all .LST or .TXT files in user specified folders.
- ✓ Determines whether an .LST file or .TXT file has a SAS compare output or is a general text file.
- ✓ If a single .LST file has multiple Compare reports (reports generated by comparing more than 1 dataset and re-directing the output to the same file), the macro loops through and checks all compare outputs.
- ✓ If a .LST file is empty (i.e. no content), the macro generates a statement on the report to indicate as such.
- ✓ The macro checks whether the datasets compared are both coming from the SAS work environment. If so, it will generate a message indicating such. The macro will also check if the datasets are from pre-defined user specified libraries.
- ✓ The macro will check whether the Production/Primary area dataset has been generated before the QC area or work area dataset as per the SOP requirements in all companies.
- ✓ The macro will check if dataset Labels match.
- ✓ The macro will check for a perfect match between the Observations and the Variables values and attributes across the 2 datasets.
- ✓ If the mismatch is very small (lower than the measurement criterion) it will consider it as a perfect match.
- ✓ The macro generates a single report which includes a summary of all findings from compare outputs in specified folders. This report can then be printed, signed and documented, thereby, eliminating the need of multiple print and signs.
- ✓ The report can be customized to fit the company requirements and/or SOP's.

IDEA BEHIND APPROACH

A general Proc Compare output generated by SAS usually will consist of the following sections:

HEADER SECTION

The header section consists of the name of the SAS procedure i.e. Proc Compare, and the name and libref of the compared datasets, along with the specified values for method and criterion options. An example is shown in Figure-1 annotation ①.

DATASET SUMMARY SECTION

The dataset summary section contains the details of the two datasets being compared. Specifically, it contains the libname, name, creation date, modification date, number of variables, number of observations and label if defined for each of the dataset. This section is always outputted by SAS irrespective of the Proc Compare results. Figure-1 ②.

VARIABLE SUMMARY SECTION

In this section, SAS reports the number of variables in common between the two datasets and also the number of variables which is in one dataset and not in the other. SAS also reports the name of the variables not in common between the two datasets provided the LISTALL options is specified by user in Proc Compare. This section also reports the variables with differing attributes viz. name, length, type and label in both datasets. Figure-1 ③.

OBSERVATIONS SUMMARY SECTION

This section contains details of the number of observations in common between the two datasets as well as the number of observations in one dataset and not in another. SAS also reports a summary of the number of observations that have equal values and unequal values for common variables. Figure-1 ④.

VALUES COMPARISON SUMMARY SECTION

SAS reports the number of variables with equal and unequal values on all compared observations. In this section, PROC COMPARE also lists limited number of unequal values for all compared variables found to be not matching. This section is only outputted in case where variables with unequal values are found in the two datasets. Figure-3 ⑦.

RESULTS SUMMARY SECTION

Although not a section on its own, SAS normally adds a note to summarize the overall result of the comparison. In case if all values compared are equal, SAS outputs a note '*No unequal values were found. All values compared are exactly equal*'. This note is not generated in case of any unequal values where the difference is greater than criterion level specified in the Proc Compare. Figure-1 ⑥.

The authors approach within the macro is to utilize and parse for messages and notes in the aforementioned sections that SAS Proc Compare procedure generates, and to programmatically determine the results of the dataset comparisons.

The COMPARE Procedure ①
Comparison of ANAL.AE with QCANAL.Q_AE
(Method=EXACT)

Data Set Summary ②

Dataset	Created	Modified	NVar	NObs	Label
PROD.AE Ⓐ	10JUN11:18:18:18 Ⓑ	10JUN11:18:18:18	49	213	AE Ⓒ
QC.AE	10JUN11:17:17:17	10JUN11:17:17:17	49	213	AE

Variables Summary ③

Number of Variables in Common: 49.

Observation Summary ④

Observation	Base	Compare
First Obs	1	1
Last Obs	213	213

Number of Observations in Common: 213.
Total Number of Observations Read from ANAL.AE: 213.
Total Number of Observations Read from QCANAL.Q_AE: 213.

Number of Observations with Some Compared Variables Unequal: 0. Ⓓ
Number of Observations with All Compared Variables Equal: 213.

NOTE: No unequal values were found. All values compared are exactly equal. ⑥

Figure 1. Annotated Proc Compare Output

MACRO REQUIREMENTS

The invocation of the macro requires the user to provide the directory paths of the folders that contain the Proc Compare outputs to be checked. The user also needs to specify the set of librefs for the directory paths where the production or primary side datasets are saved. In our macro, these directory paths are assigned to individual macro references/variables. The directory paths are provided as a string of this macro references or libnames separated by spaces at macro invocation. The macro searches for these user defined libname to determine the name, time of creation and the location of production dataset and QC dataset listed the dataset summary section.

PROGRAMMING LOGIC

The macro loops through each directory paths/libnames defined by user as containing Proc compare output and reads in the filenames and timestamp details of.TXT or .LST files located within. The macro then parses through each filename and reads in the contents of the file into a single variable dataset. Each observation in the dataset corresponds to each line in the file. Provision has been made within the macro to automatically adjust the line size to the longest line in the file. Having imported the contents of this file into this single variable dataset, the macro will start checking the different scenarios to determine the results of the Proc Compare. The macro requires the LISTALL option to be specified in Proc Compare in case the user wants to lists the name of uncommon variables in the report.

The macro assumes that the Proc Compare results are outputted only as a .lst or .txt file and only looks for these file extensions while parsing files in a folder. The macro only reports files that are totally blank or files identified as containing Proc Compare output. The macro searches for the strings 'The Compare Procedure' and 'Data Set Summary' in the file to check if the file contains any Proc Compare output.

The next section summarizes some of the most common Proc Compare results/scenarios and manual quality control checks and provides the reader with logic used to programmatically determine these results.

'ALL EQUAL' SCENARIOS

Scenario 1: When all values, variables and observations are exactly equal.

In this case, SAS outputs a string in the results summary section as shown by annotation ⑥ in Figure 1. The macro searches for this string and reports overall results as 'ALL EQUAL' as shown in test case 01 in report.

Scenario 2: When all values, variables and observations are exactly equal but values are differing by a small amount greater than the set criterion.

```
Values Comparison Summary

Number of Variables Compared with All Observations Equal: 67.
Number of Variables Compared with Some Observations Unequal: 0.
Total Number of Values which Compare Unequal: 0.
Total Number of Values not EXACTLY Equal: 18.
Maximum Difference Criterion Value: 2.0955E-16.
```

Figure 2. Example Output with Small Difference in Numeric Variable

In this case, the above string is not outputted as shown in figure 2. In such cases, to determine if all values compared are equal, we also need to check for existence of string '*Number of Observations with Some Compared Variables Unequal: 0.*' under observation summary and '*Number of Variables Compared with Some Observations Unequal: 0.*' under value comparison summary section. The overall results are reported as 'ALL EQUAL' as shown in test case 02.

Scenario 3: When all values and observations are exactly equal but number of variables differ.

In such cases, SAS output a string 'Number of variables in #Plibname.dataset and not in #Qlibname.dataset' and macro searches for this string to determine if there any uncommon variables. The libname and dataset in above string is extracted by the macro from the dataset summary section. If LISTALL option has been specified, in addition to the above string, SAS also lists the names of variables in one dataset and not in another. The macro utilizes the user defined production library names and the SAS generated list of variables to report the variable names in production dataset and not in QC dataset as shown in test case 03 in report. The macro currently does not list the variable names in QC dataset and not in production dataset but can be easily be modified to do the same. Since all

values that were compared are equal, the macro reports the overall results as 'ALL EQUAL' and lists the uncommon variables in extra variables summary column.

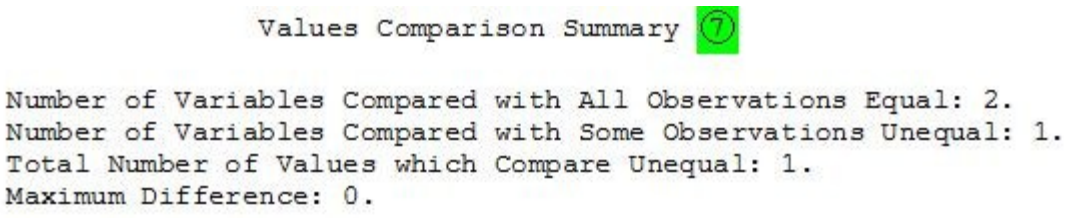

Figure 3. Example Output when Value Differ

'SOME UNEQUAL' SCENARIOS

Scenario 4: When all values and variables are equal but number of observations differ.

Incase if the number of observations are different between the two datasets than the overall results if reported as 'SOME UNEQUAL' as shown in test case 09 in report and the reason is provided as 'Number of observations in compared datasets not equal'. The number of observations in each dataset is derived by the macro from the number reported in the dataset summary section. The reader should note that even if all conditions for 'ALL EQUAL' scenarios listed above are met but if the numbers of observations in the two datasets are different, the result is reported as 'SOME UNEQUAL'.

Scenario 5: Variable values differ.

If the combination of conditions for equality in Scenario 1 and 2 are not met, then macro reports the overall results as 'SOME UNEQUAL' and the non-matching reason is listed as 'Variables or observations found with unequal values' as shown in test case 05 in report.

Scenario 6: When all values, variables and observations are exactly equal but variable attributes differ.

SAS outputs a note 'Variables with Differing attributes' in variable summary section in case of any variable found with mismatching attributes. The macro checks for this string 'differing attributes' and in such scenario the overall results is reported as 'SOME UNEQUAL' and the reason provided as 'Some variables found with differing attributes in compared Datasets' as shown in test case 07.

CHECKS

Check 1: Production side dataset is run before the QC dataset.

The macro uses the user defined production libnames and the SAS reported date and times of the two dataset Figure-1 Ⓑ in dataset summary section to perform this check. In case if the datasets were run in the wrong order then the macro reports the reason as 'QC Dataset was run before production side as shown in test case 04 in report.

Check 2: Datasets labels are the same.

The macro utilizes the datasets label reported by SAS in the dataset summary section Figure-1 Ⓒ to perform this check and the results are reported as 'Compared Datasets labels don't match'.

Check 3: One dataset is from user defined production environment.

Dataset	Created	Modified	NVar	NObs
WORK.ORIGINAL	05OCT12:16:03:19	05OCT12:16:03:19	3	104
WORK.QC_T_DEM001	05OCT12:16:03:19	05OCT12:16:03:19	3	104

Figure 4. Both datasets are from Work library.

The macro reads in compared datasets names from the dataset summary section Ⓐ and checks if libname of one of the dataset is in user specified production library names. The results of the check are reported as either 'Both compared datasets are from work folders' OR 'Both compared datasets are from defined production folders' OR 'Both compared datasets are from undefined production folders' as shown in test case 06.

Check 4: Blank compare output.

There can be instances when the Proc Compare fails and as a result generates an empty output file. The macro checks for such files and report as shown in test case 10 in report.

FINAL REPORT AND INTERPRETATION

LST File/ Compare Dataset Name	Overall Result	Extra Variables Summary	Non-matching Reasons QC Check Result	Comment and Initials
tc01_allmatch.lst 1-ANAL.AE	ALL EQUAL			
tc02_allmatch_small_diff.lst 1-DMCANAL.ADSL	ALL EQUAL			
tc03_allmatch_extravars.lst 1-DATATAB.T_AECTC_DRGWITH	ALL EQUAL	Variables in Production but not in QC- ORD_PG ORD1		
tc04_allmatch_wrongorder.lst 1-PROD.DM_ATTRB	ALL EQUAL		QC DATASET WAS RUN BEFORE PRODUCTION	
tc05_nonmatch.lst 1-PROD.DM	SOME UNEQUAL		VARIABLES FOUND WITH UNEQUAL VALUES	
tc06_allmatch_bothwork.lst 1-WORK.ORIGINAL	ALL EQUAL		BOTH COMPARED DATASETS ARE FROM WORK FOLDERS	
tc07_allmatch_diff_attrib.lst 1-PROD.DM	SOME UNEQUAL		SOME VARIABLE ATTRIBUTES DIFFER IN COMPARED DATASETS	
tc08_allmatch_undefinedlib.lst 1-ANAL_U.AE	ALL EQUAL		BOTH COMPARED DATASETS ARE FROM UNDEFINED PRODUCTION FOLDERS	
Tc09_allmatch_diff_obs.lst 1-PROD.DM	SOME UNEQUAL		NUMBER OF OBSERVATIONS IN COMPARED DATASETS NOT EQUAL	
tc10_empty_file.lst			FILE IS COMPLETLY BLANK.PLEASE CHECK	

Reviewed by: Date: Page 1 of 1

Figure 5. Macro Report Example

The report will display results for all lst and txt files from all directory paths specified by the user during macro invoke with each new directory path starting on a new page with the directory name displayed at the top of the page. We have provided an example report generated by the macro listing the results of Proc compare outputs from one directory path. We have also explained the report layout below briefly describing the contents and purpose of each column.

Column 1: LST File and compared dataset name

The macro reports the name of the Proc Compare output file as well the dataset name from user specified production environment. If both compared datasets are not from any of the specified production libraries, the macro will report the name of the first dataset in the Data Summary section as shown in test case 06. In case of multiple Proc Compare outputs in the same file, the macro will list the results of these various outputs under the same file name but with their respective dataset names.

Column 2: Overall result

The overall results of the datasets comparison as determined by macro i.e. ‘ALL EQUAL’ and ‘SOME UNEQUAL’ are listed under this column. As described in programming logic, ‘ALL EQUAL’ results signifies that the values for all compared variables are exactly equal or numeric variable values are differing by an inconsequential value. The ‘SOME UNEQUAL’ values are reported when the some differences are observed in compare variable values or attributes or in the number of observations.

Column 3: Extra Variable Summary

This column will list the names of the variables in the production environment dataset and not in QC dataset. The requirements for this column may differ based upon each company need or SOP. The macro can easily be modified to either suppress the column or report uncommon variables in both datasets to meet process needs.

Column 4: Non Matching Reason\QC Check Result.

The reason for the dataset comparison to be considered as non-matching i.e ‘SOME UNEQUAL’ and/or the results of the checks are listed in this column. The results of the QC checks are displayed irrespective of the results of the proc comparison derived by the macro. This column has been planned such that if any reason or QC check result is

reported for a Proc compare output, than the compare result is not deemed as passed QC and needs to be investigated further. This layout was decided keeping the convenience of the reviewer in mind as the reviewer has to just ensure that this column is blank in order to ratify the results of the Proc Compare.

Column 5: Comments and Initial

The column has been provided for reviewer to provide comments for issues reported in compare results that cannot be fixed or found to have no consequence on the deliverable.

CONCLUSION

Independent double programming is the preferred method of validation for all TFL's in most companies. Proc Compare is the SAS procedure used to compare the independently programmed SAS datasets. The Proc Compare typically generates an .LST file which contains the results of comparison. Many companies are still using a manual method of checking the SAS compare output .LST files to evaluate the compare results. This method involves opening each of these compare output files and visually checking them or it involves printing each of these compare outputs and, ticking and signing on each page. This method is extremely time-consuming, cumbersome and error-prone.

The LST Compare macro loops through all compare output files in all user specified directories, checks for the existence of mismatches, performs common manual QC checks, and summarizes the findings in a report. The method automates the entire print, tick and sign process by reading in the SAS compare report output and checking for the necessary details. The macro generates a single report which is organized in an intuitive way for the reviewer to understand the details of the compare.

The macro's benefits are not just limited to the programming personnel but extend beyond to the department and the company as well. The time savings are huge and can save up to several hours per delivery. On a cumulative basis this amounts to potential savings of several thousand hours per year which correlates to hundreds of thousands of dollars in savings. This tremendous cost saving and resource re-utilization is the foremost advantage of this method. In addition to major cost savings, this automated method combined with the intuitively planned report, will considerably reduce or even eliminate the error rate associated with the manual process and will require less training and oversight. The downstream cost, effort and time related to paper filing is also significantly reduced as the macro generates a single report for more effective filing and documentation. Overall this macro is a very useful and much needed tool for any company still following the process of manually checking, printing, signing and documenting each individual Proc Compare output.

CONTACT INFORMATION

SANKET KALE
sanketkale@gmail.com

SAJIN JOHNNY
sajinjohnny@gmail.com

Source Code Sample

SAMPLE CODE 1 : READING IN LST FILES FROM SPECIFIED UNIX DIRECTORY

```
***********USING UNIX COMMANDS TO READ IN LST FILES FROM ONE DIRECTORY***
************AND OUTPUT TO EXTERNAL TEXT FILE **************************;
 DATA _null_;
      CALL system("cd &&flpath&i");
      CALL system("ls -l *.lst > lstfiles.txt");
 RUN;

******EXSTRACTING LST FILE NAME, FILESIZE, FILE CREATION DATETIME FROM THE***
***********THE OUTPUTTED TEXT FILE FROM UNIX*******************************;

 DATA compfile;
      infile "&&flpath&i..lstfiles.txt" length=reclen;
      length day $2 month $3 year $4 flsize $10  ;
      inPUT @1 record $varying200. reclen;
      IF index(upcase(record), ".LST");
      flname=strip(scan(record,-1,' '));
      year  =strip(scan(record,-2,' '));
      IF index(year,':') THEN year=strip(PUT(year("&sysdate."d),best.));
      day   =strip(scan(record,-3,' '));
      month =strip(scan(record,-4,' '));
      IF input((day||month||year),date9.)>"&sysdate."d
         THEN year=strip(PUT(inPUT(year,best.)-1,best.));

      fldate=day||month||year;
      flsize=strip(scan(record,-5,' '));
 RUN;

***************ASSIGNING NAME OF EACH READ IN LST FILE FROM DIRECTORY*******
***************TO MACRO VARIABLE*****************************************;
 DATA _null_;
     set compfile end=eof;
     CALL symPUT("flname"||strip(PUT(_n_,best.)),strip(flname));
     IF eof THEN CALL symPUT("nflname",strip(PUT(_n_,best.)));
 RUN;
```

SAMPLE CODE 2 : DETERMINING LIBRARY OF COMPARED DATASETS AND RUN ORDER

```
DATA dsetsum2;
  set dsetsum1 ;
  format a_dttm b_dttm datetime18.;
  length a_label b_label $100.;
  ******EXTRACTING LIBNAME, DATASET NAME, DATETIME, NO OF OBS AND LABEL OF
  SECOND DATASET****;

  b_lib  = scan(scan(col2,1,' '),1,'.');
  b_name = scan(scan(col2,1,' '),2,'.');
  b_dttm = inPUT(scan(col2,2,' '),datetime16.);
  b_obs  = inPUT(scan(col2,5,' '),best.);
```

```
  IF lablpos > 0 THEN b_label=substr(col2,lablpos);

 call symput("FRSTNAM",strip(a_lib)||'.'||strip(a_name));
 call symput("NUMOBS",strip(put(a_obs,best.)));

*******CHECKING IF BOTH DATASET ARE FROM WORK FOLDERS AND SETTING MACRO
VARIABLE VALUE***;
 if indexw(upcase(a_lib),"WORK") and indexw(upcase(b_lib),"WORK") then
    call symput("DSETSUMM","WW");
*******CHECKING IF BOTH DATASET ARE FROM PRODUCTION FOLDERS AND SETTING MACRO
VARIABLE VALUE***;
 else if indexw("&PRODLIBS", upcase(a_lib)) and indexw("&PRODLIBS",
 upcase(b_lib)) then
    call symput("DSETSUMM","PP");
***********CHECKING IF BOTH DATASET ARE FROM UNDEFINED FOLDERS AND SETTING
MACRO VARIABLE VALUE***;
 else if indexw("&PRODLIBS", upcase(a_lib))=0 and indexw("&PRODLIBS",
 upcase(b_lib))=0 then
    call symput("DSETSUMM","NN");

*****CHECKING IF FIRST DATASET IS FROM PRODUCTION FOLDER AND SECOND FROM*****
*****NON PRODUCTION LIBRARY. IF YES, CHECKING IF FIRST DATASET WAS RUN  *****
*****BEFORE THE SECOND DATASET AND ASSIGN LIBNAME AND DATASET NAME OF *******
*****DETERMINED PRODUCTION AND QC DATASET TO MACRO VARIABLE********;

 else if indexw("&PRODLIBS", upcase(a_lib)) and indexw("&PRODLIBS",
 upcase(b_lib))=0 then
    do;
      call symput("DSETSUMM","PQ");
      if a_dttm > b_dttm > . then call symput("DTTIME","WRONG");
      else if b_dttm > a_dttm > . then call symput("DTTIME","RIGHT");

      call symput ("PLIB",strip(upcase(a_lib)));
      call symput ("QLIB",strip(upcase(b_lib)));

      call symput ("PNAM",strip(upcase(a_name)));
      call symput ("QNAM",strip(upcase(b_name)));

    end;

*****CHECKING IF SECOND DATASET IS FROM PRODUCTION FOLDER AND SECOND FROM****
*****NON PRODUCTION LIBRARY. IF YES, CHECKING IF SECOND DATASET WAS RUN  ****
*****BEFORE THE FIRST DATASET AND ASSIGN LIBNAME AND DATASET NAME OF *******
******DETERMINED PRODUCTION AND QC DATASET TO MACRO VARIABLE********;


  else if indexw("&PRODLIBS",upcase(a_lib))=0 and
   indexw("&PRODLIBS",upcase(b_lib)) then
    do;
      call symput("DSETSUMM","QP");
      if a_dttm > b_dttm > . then call symput("DTTIME","RIGHT");
      else if b_dttm > a_dttm > . then call symput("DTTIME","WRONG");

      call symput ("QLIB",strip(upcase(a_lib)));
      call symput ("PLIB",strip(upcase(b_lib)));

      call symput ("QNAM",strip(upcase(a_name)));
```

```
      call symput ("PNAM",strip(upcase(b_name)));
    end;

  **********CHECKING IF LABEL OF BOTH DATASET MATCH AND ASSIGN TO*******
  *************** MACRO VARIABLES ******;

  if a_label = b_label then call symput("DSETLABL","MATCH");
   else call symput("DSETLABL","NOMATCH");

  ********CHECKING IF NO OF OBS OF BOTH DATASET MATCH AND ASSIGN ********
  ***********TO MACRO VARIABLES ******;

  if a_obs = b_obs then call symput("DSETOBS","EQUAL");
   else call symput("DSETOBS","UNEQUAL");
;

RUN;

**********READING IN VARIABLE SUMMARY SECTION AND CHECKING IF ANY *******;
***********VARIABLES IN ONE DATASET AND NOT IN ANOTHER**********;

%IF &DSETSUMM=PQ or &DSETSUMM=QP %THEN %DO;
 DATA varsum1(drop=_counter _group);
   set lstcont2(WHERE=(_counter=&K and _group=2)) end=eof ;
   length VARPNOTQ VARQNOTP $300;
   retain FLAGPQ TYPPOS VARPNOTQ VARQNOTP;

   IF index(contnt,"NUMBER OF VARIABLES IN &PLIB..&PNAM BUT NOT IN
   &QLIB..&QNAM")
   THEN call symput("INPNOTQ","Y");

   IF index(contnt,"NUMBER OF VARIABLES IN &QLIB..&QNAM BUT NOT IN
   &PLIB..&PNAM")
   THEN call symput("INQNOTP","Y");

   IF index(contnt,"LISTING OF VARIABLES IN &PLIB..&PNAM BUT NOT IN
   &QLIB..&QNAM")
   THEN FLAGPQ="P";
   IF index(contnt,"LISTING OF VARIABLES IN &QLIB..&QNAM BUT NOT IN
   &PLIB..&PNAM")
   THEN FLAGPQ="Q";
***************CHECKING IF ANY OF THE COMPARED VARIABLES HAVE DIFFERING****
*************** ATTRIBUTES************************************;
   IF index(contnt,"DIFFERING ATTRIBUTES") THEN
   do;
    FLAGPQ=" ";
    call symput("DIFFATTRB","Y");
   end;

   IF FLAGPQ='P' and index(compress(contnt,' '),'VARIABLETYPELENGTH') then
    TYPPOS=index(contnt,'TYPE');
   IF FLAGPQ='Q' and index(compress(contnt,' '),'VARIABLETYPELENGTH') then
    TYPPOS=index(contnt,'TYPE');
****** READING IN NAMES OF VARIABLES IN ONE PRODUCTION DATASET*******
******AND NOT IN OTHER DATASETAND ASSIGNING THE NAMES SEPARATED BY SPACE TO**
******MACRO VARIABLE********************;
```

```
   IF TYPPOS ne . THEN
   do;
    IF FLAGPQ='P' and substr(contnt,TYPPOS,3) in ('CHA','NUM') and
     input(scan(contnt,3,' '),best.) > 0 THEN
      VARPNOTQ=strip(VARPNOTQ)||' '||strip(SCAN(CONTNT,1,' '));
    IF FLAGPQ='Q' and substr(contnt,TYPPOS,3) in ('CHA','NUM') and
     input(scan(contnt,3,' '),best.) > 0 THEN
      VARQNOTP=strip(VARQNOTP)||' '||strip(SCAN(CONTNT,1,' '));
   end;

   IF eof THEN
    do;
    call symput("VARPNOTQ",strip(varpnotq));
    call symput("VARQNOTP",strip(varqnotp));
    end;
RUN;

************** END OF SAMPLE CODE ***************;
```