Paper 76505-2011

"Compare Me" a SAS® Datasets Comparison Tool

Anurag Katare, Lake Hiawatha, NJ
Jayesh Soneji, Princeton, NJ

Abstract

SAS® Datasets comparison tool called "Compare-Me" is a tool built to facilitate programmers to compare 'n' pairs of datasets located in two different directory locations. The comparison differences between the datasets are then documented and are presented in an excel sheet. This tool helps compare CSV and EXCEL files using the Proc Compare SAS procedures. This tool also gives the user a feature to provide all the 'by' variables as keys that would be used for comparison of two datasets.

Introduction

This paper provides details of initial setup that is required for the tool and also goes through a step by step process on how to utilize the SAS Compare Me tool. This tool had been created during the testing phase of a large Life Sciences and Commercial operations project at one of the top pharmaceutical company. The tool was created with a view to support the comparison of the datasets in the old vs. the new environments, during an upgrade project and was heavily utilized during the System Integration testing (SIT) and User acceptance testing (UAT) phase of the project. It greatly helped in reducing the SIT and UAT testing times, and assisting the testing team to test approximately 2300 SAS datasets thereby resulting in huge time savings of at lest ~40 person days during the course of the project.

The tool is automated to an extent that once it is setup correctly, there is no manual intervention required. Behind the scenes, "Compare Me" tool runs the Proc Compare SAS procedure. It reduces the unnecessary overhead of writing a comparison code every time some datasets needs to be compared.

With this tool two output files are obtained. The first output file, a CSV file contains brief description about comparison results for:

- Datasets completely matched
- Data mismatch
- Format mismatch
- Data type mismatch etc.

The second output file, a PDF file contains detailed comparison result that are generated by Proc Compare.

Benefits of "Compare Me" Tool

- Easy to use. The user of the tool needs to enter the details in .CSV file. These details are passed as a parameter to the SAS Code.
- Required "by variables" used as keys to compare the datasets are driven from a metadata table.
- Can compare 'n' pairs of SAS datasets at a time. (any number of SAS datasets depending on the server capacity)
- Limited user intervention is required. "Compare Me" tool is executed in the background on the UNIX server and requires minimal user intervention.
- Generates the result of comparison of datasets in .CSV file format and the sequence of SAS datasets compared is maintained. Results sheet provides brief result of comparison, such as "matched", "un-match", "Data type mismatch" , "Length mismatch" etc.
- The tool also generates detailed outputs of each pair of datasets comparison in separate PDF file
- No knowledge of SAS is required for using this tool.
- The tool uses PROC COMPARE procedure to compare datasets.

Initial Setup

This package has been designed primarily for SAS on a UNIX environment. The same SAS programs can be used with SAS on Windows with minor tweaks in the code.

To use the SAS code in UNIX, one needs to follow the following instructions:

- Store the below listed SAS code and Shell Scripts in the same Unix directory path:
    - Get_Metadata.sas (details below)

- Get_Metadata.sh
- Dataset_Comparison.sh &
- Dataset_Comparison.sas
- The Get_Metadata.sas SAS code can be very useful if you are not going for full dataset comparisons. The only update required is in the shell script providing input file path and path to store temporary SAS dataset.

- Get_metadata.sas provides metadata information to users if they need information about the datasets being compared. For example, it gives information about record count, variable names, data type, etc.
- Get_metadata.sas facilitates users when they want to customize their data comparison. The user can have the capability to compare only some selective columns then it can be directly copied from the output.csv file of get_metadata.sas.
- Next step is to keep "Input.csv" file in one directory. It can be the same directory where codes are present but a separate Input/output is preferable. This is the input file that we need to fill before comparison starts.

Get Metadata Information

The purpose of getting the Metadata information is to help the user access all the variables that are available in both the datasets that is required to be compared. The user can then decide the variables he would like to compare and select those variables for comparison.

The user can see some key metadata information of the SAS datasets like the variable names, data type, and label of the datasets that is required for comparison. The user can also find out the by variables required for sorting the dataset before comparing.

The metadata information is also used to update parameters in get_metadata.sh shell script. Some of the parameters are:

- Path of the datasets
- Output Path and file name of metadata .csv

Syntax:
-sysparm <dataset location>~<metadata CSV file name with location>~<Columns CSV files with location>

The first output file which details the metadata information of the datasets is shown below with the columns:

First output file is metadata file will list out details of datasets shown below.

MEMNAME	NAME	TYPE	LENGTH	VARNUM	LABEL	NOBS

MEMNAME: Provides the dataset name
NAME: Provides the variable name
TYPE: Provides the data type of the variable
VARNUM: Provides the variable number for number type of variables
NOBS: - Provides the count of number of observations in the SAS dataset

Second output file is column CSV file will list out 1 row for each dataset and all the variable names in one row separated by space.

MEMNAME	COLUMN_NAME	DATASET_PATH

MEMNAME: Provides the dataset name
COLUMN_NAMES: Provides variable name
DATASET_PATH: Provides the dataset path

Required Inputs for tool

This tool requires the Input.csv file to be filled to start comparing the two datasets. Following list of columns are available in the Input.csv with details.

1. CMP_FORMAT:
   Allows the user to provide file format (e.g SAS, CSV, XLS).

2. CMP_COLUMN:

Allows the user to provide value as “ALL” or allows users to provide list of SAS Dataset variable name separated by space. “ALL” will by default take all the SAS Dataset variables for comparison

3. CMP_ROWS:
   Allows the user to provide “ALL” or allow entering number of rows to be compared. “ALL” will by default compare all the rows. For example if the number 100 is given, only the first 100 records of both datasets will be compared.
4. CMP_FILE1:
   Allows the user to provide path of the first dataset that is required to be compared.

5. CMP_FILE2:
   Allows the user to provide path of the second dataset that is required to be compared.

6. CMP_FILENM1:
   Allows the user to provide name of first dataset that is required to be compared.

7. CMP_FILENM2:
   Allows the user to provide name of second dataset that is required to be compared.

8. CMP_OUTPUT:
   Allows the user to provide the output PDF file name with location. This output is the one that we get as an output of PROC COMPARE.

9. Sort_Required:
   Allows the user to provide either “YES” or “NO”. “Yes”, if the user wants to sort datasets else specify “NO”.

10. Sort_Columns1:
    Allows the user to provide preference of sorting the dataset. Leave blank if sorting is not required else give variable name that can be used as BY Variable for the first dataset.

11. Sort_Columns2:
    Allows the user to provide preference of sorting the dataset. Leave blank if sorting is not required else give variable name that can be used as BY Variable for second dataset.
12. Temp_area:
    Give a path to be used as a temporary area to create temporary datasets. Program itself deletes all the datasets created in temp area before completion.

13. Filter_Required:
    Allows the user to provide either “YES” or “NO”. “YES”, if you want to compare filtered data of both datasets else specify “NO”

14. Where:
    Allows the user to specify a filter condition for example EMP_ID = 10234 AND EMP_NAME = ‘ANURAG’. Leave it blank if value of “Filter_Required” field is NO.

15. Colum_map_required:
    Allows the user to provide ether “YES” or “NO”. “YES”, if variable names of both the datasets are not same else specify “NO”.
    Note: If variable names are not same then give the variable names of first dataset in “CMP_COLUMN” field and second dataset in the following field.

16. Mapping_colums_of_DS2:
    If “colum_map_required” is “NO” then leave blank else give variable names in second datasets separated by space. Note: Variable order plays a key role here because order specified here maps the variable order specified in “CMP_COLUMN” field for first dataset.

17. No_of_differences_printed:
    This field is to limit number of differences listed in CSV file (i.e. detailed_differences_1.csv).

Details of Output

Outputs are categorized as Metadata Output and Comparison Output. Metadata Output is the output of get_metadata.sas program and Comparison Output is the output of Dataset_Comparison.sas program.

Metadata Output

Following are two outputs of metadata program to provide users information about datasets if required before they compare it. It can be very useful when you want to compare datasets having different variable names. In this case user can easily copy the variable names to fill main input file for comparison.

1. **Columns.csv:** This output file name can be different as specified in get_metadata.sh script. Each row of this file gives dataset name in first column, all the variable name separated by space in second column and data location in third column. It lists all the datasets available in the path specified in shell script before execution.

MEMNAME	COLUMN_NAME	DATASET_PATH
TEST_01	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_02	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_03	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_04	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_05	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_06	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_07	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_08	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_09	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_10	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_11	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_12	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_13	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_14	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_15	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_16	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_17	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_18	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_19	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO
TEST_20	ID FIRST_NAME LAST_NAME DEPT	/EMP/INFO

2. **Metadata.csv:** This output file name can be different as specified in get_metadata.sh script. Instead of giving one row for each dataset this file gives one row for each variable of dataset and provides more details of that variable. First column specifies dataset name, second column specifies variable, name third column specifies data type 1 for numeric 2 for character, forth column is length, fifth is variable number as per the order in dataset, sixth is label and seventh column specifies number of observation in the dataset. This output file lists details of all the datasets available in the path specified in shell script before execution.

MEMNAME	NAME	TYPE	LENGTH	VARNUM	LABEL	NOBS
ABC	DSPND_MTH_END_DATE	1	8	1	DATA MONTH END DATE	11248
ABC	MKT_DESC	2	40	2	MARKET NAME	11248
ABC	DCL_TC_CD	2	3	3	TC CODE	11248
ABC	DCL_SEG_CD	2	12	4	SEGMENT CODE	11248
ABC	DCL_CAT_CD	2	12	5	CATEGORY CODE	11248
ABC	DCL_PROD	2	12	6	PRODUCT CODE	11248
ABC	CORP	2	15	7	ASSOCIATED CORPORATION	11248
ABC	DRUG_NAME	2	60	8	DRUG NAME	11248
ABC	BRND_GEN_IND	2	1	9	BRAND GENERIC INDICATOR	11248
XYZ	QTY	1	8	1	QUANTITY	12005
XYZ	PYMT_TYPE	2	7	2	PAYMENT TYPE	12005
XYZ	DCL_PLAN_AMC_ID	2	6	3	AMC PLAN Id	12005
XYZ	DCL_PBM_AMC_ID	2	6	4	DCL_PBM AMC Id	12005
XYZ	DCL_PBM_AMC_NAME	2	60	5	DCL_PBM AMC NAME	12005
XYZ	NATN_INSR_AMC_ID	2	6	6	NATIONAL INSURANCE AMC ID	12005
XYZ	STATE	2	2	7	STATE	12005
XYZ	MSA_CD	2	5	8	PHARMACY MSA CODE	12005
XYZ	REFILL_IND	2	1	9	REFILL INDICATOR	12005

Comparison Output

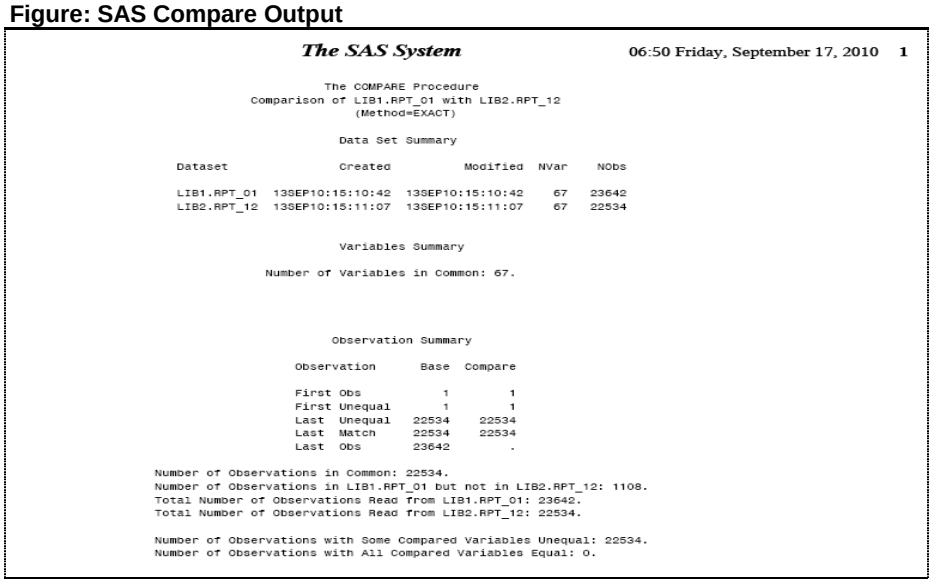
1. **Result (.csv file)**
   The result file provides the result of the SAS dataset compared. It provides the location of the compared files, file name and the result of the two datasets compared.

CMP_FILE1	CMP_FILE2	CMP_FILENM1	CMP_FILENM2	RESULT
/emp/info/dp1	/emp/info/dp2	emp_01	emp_02	Data not matched, first dataset has more rows than second.
/emp/info/dp1	/emp/info/dp2	emp_03	emp_04	Matched successfully

2. Proc Compare Output (In PDF):

The result of the PROC COMPARE details the compared results between the two dataset. Following is the screen shot of the output.

Figure: SAS Compare Output



The SAS System 06:50 Friday, September 17, 2010 1

The COMPARE Procedure
Comparison of LIB1.RPT_01 with LIB2.RPT_12
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
LIB1.RPT_01	13SEP10:15:10:42	13SEP10:15:10:42	67	23642
LIB2.RPT_12	13SEP10:15:11:07	13SEP10:15:11:07	67	22534

Variables Summary

Number of Variables in Common: 67.

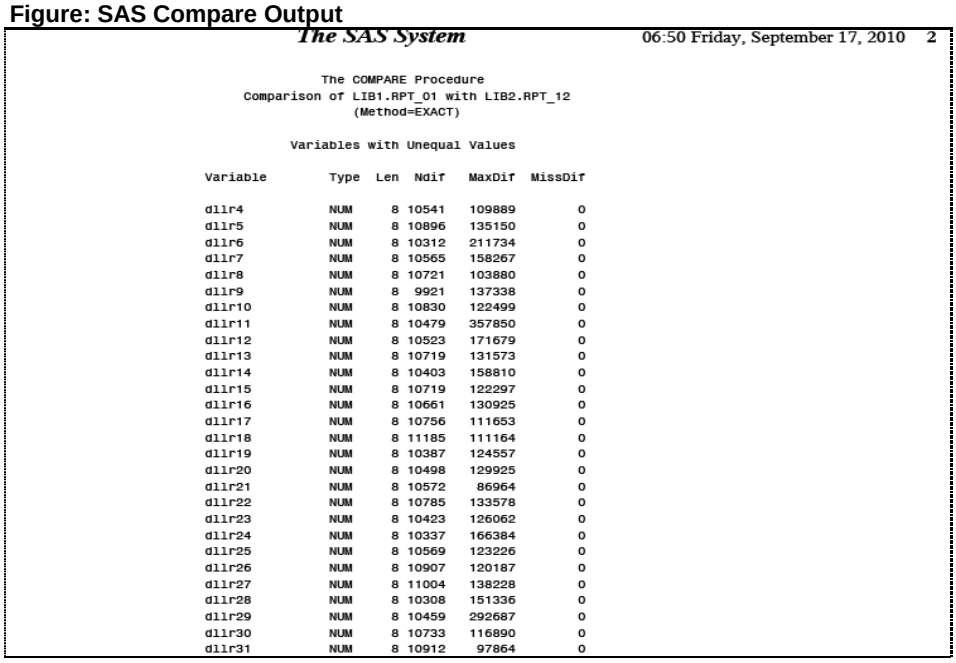
Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	22534	22534
Last Match	22534	22534
Last Obs	23642	.

Number of Observations in Common: 22534.
Number of Observations in LIB1.RPT_01 but not in LIB2.RPT_12: 1108.
Total Number of Observations Read from LIB1.RPT_01: 23642.
Total Number of Observations Read from LIB2.RPT_12: 22534.

Number of Observations with Some Compared Variables Unequal: 22534.
Number of Observations with All Compared Variables Equal: 0.

Figure: SAS Compare Output



The SAS System 06:50 Friday, September 17, 2010 2

The COMPARE Procedure
Comparison of LIB1.RPT_01 with LIB2.RPT_12
(Method=EXACT)

Variables with Unequal Values

Variable	Type	Len	Ndif	MaxDif	MissDif
dllr4	NUM	8	10541	109889	0
dllr5	NUM	8	10896	135150	0
dllr6	NUM	8	10312	211734	0
dllr7	NUM	8	10565	158267	0
dllr8	NUM	8	10721	103880	0
dllr9	NUM	8	9921	137338	0
dllr10	NUM	8	10830	122499	0
dllr11	NUM	8	10479	357850	0
dllr12	NUM	8	10523	171679	0
dllr13	NUM	8	10719	131573	0
dllr14	NUM	8	10403	158810	0
dllr15	NUM	8	10719	122297	0
dllr16	NUM	8	10661	130925	0
dllr17	NUM	8	10756	111653	0
dllr18	NUM	8	11185	111164	0
dllr19	NUM	8	10387	124557	0
dllr20	NUM	8	10498	129925	0
dllr21	NUM	8	10572	86964	0
dllr22	NUM	8	10785	133578	0
dllr23	NUM	8	10423	126062	0
dllr24	NUM	8	10337	166384	0
dllr25	NUM	8	10569	123226	0
dllr26	NUM	8	10907	120187	0
dllr27	NUM	8	11004	138228	0
dllr28	NUM	8	10308	151336	0
dllr29	NUM	8	10459	292687	0
dllr30	NUM	8	10733	116890	0
dllr31	NUM	8	10912	97864	0

3. Detailed Difference (detailed difference in .csv file)

The detailed difference .csv file captures all the differences between the columns that are being compared. The .csv file lists down all the differences between the columns being compared of the two datasets.

TYPE	_OBS_	Client_Number	Report_Number	SRA1	SRA2	SRA3	SRA4	Category_Code	Product_Group	Date	dllr1	dllr2	dllr3	dllr4	dllr5	dllr6
BASE	1	60	1		A2	935253	HOSPITAL	1	0	3-Sep-10	0	0	0	0	0	0
COMPARE	1	60	12		A2	43223252		1	0	3-Sep-10	3294.4	3294.4	0	3294.4	0	3294.4
DIF	1	...	XX	.....	.....	XXXXX..X..	XXXXXXXX..	..	.....	E	3294.4	3294.4	E	3294.4	E	3294.4
BASE	2	60	1		A2	935253	HOSPITAL	1	3	3-Sep-10	0	0	0	0	0	0
COMPARE	2	60	12		A2	43223252		1	2	3-Sep-10	0	0	0	0	0	0
DIF	2	...	XX	.....	.....	XXXXX..X..	XXXXXXXX..	..	X	E	E	E	E	E	E	E
BASE	3	60	1		A2	33426250	HOSPITAL	1	0	3-Sep-10	0	2	0	0	0	0
COMPARE	3	60	12		A2	43223252		1	3	3-Sep-10	3294.4	3294.4	0	3294.4	0	3294.4
DIF	3	...	XX	.....	.....	X.X.X..X..	XXXXXXXX..	..	X	E	3294.4	3292.4	E	3294.4	E	3294.4
BASE	4	60	1		A2	33426250	HOSPITAL	1	3	3-Sep-10	0	2	0	0	0	0
COMPARE	4	60	12		A2	43223252		1	4	3-Sep-10	3294.4	3294.4	0	3294.4	0	3294.4
DIF	4	...	XX	.....	.....	X.X.X..X..	XXXXXXXX..	..	X	E	3294.4	3292.4	E	3294.4	E	3294.4
BASE	5	60	1		A2	43223252	HOSPITAL	1	0	3-Sep-10	0	0	0	0	0	0
COMPARE	5	60	12		A3	2451251		1	0	3-Sep-10	2200.1	2200.1	0	2200.1	0	0
DIF	5	...	XX	.....	.X.....	XXXXX..X..	XXXXXXXX..	..	.....	E	2200.1	2200.1	E	2200.1	E	E
BASE	6	60	1		A2	43223252	HOSPITAL	1	1	3-Sep-10	0	0	0	0	0	0
COMPARE	6	60	12		A3	2451251		1	2	3-Sep-10	2200.1	2200.1	0	2200.1	0	0
DIF	6	...	XX	.....	.X.....	XXXXX..X..	XXXXXXXX..	..	X	E	2200.1	2200.1	E	2200.1	E	E
BASE	7	60	1		A2	43223252	HOSPITAL	1	3	3-Sep-10	0	0	0	0	0	0
COMPARE	7	60	12		A3	2451251		1	3	3-Sep-10	0	0	0	0	0	0
DIF	7	...	XX	.....	.X.....	XXXXX..X..	XXXXXXXX..	..	.....	E	E	E	E	E	E	E
BASE	8	60	1		A3	2451251	HOSPITAL	1	0	3-Sep-10	0	0	2	0	0	0
COMPARE	8	60	12		A3	2451251		1	4	3-Sep-10	2200.1	2200.1	0	2200.1	0	0
DIF	8	...	XX	.....	.....	.....	XXXXXXXX..	..	X	E	2200.1	2200.1	-2	2200.1	E	E

Programming Details (Sample Code)

There are two SAS programs written for this tool Get_metadata.sas and Dataset_Comparison.sas.
Basic SAS procedures are used in these programs, below are the brief details of those procedures used.

Get_metadata.sas

PROC CONTENTS is used to extract details of datasets like variable names, data type, length etc.

```
PROC CONTENTS DATA=SASDATA._ALL_ MEMTYPE=DATA
OUT= REF (KEEP=MEMNAME NAME TYPE LENGTH VARNUM LABEL NOBS) NOPRINT;
RUN;
```

PROC SQL is used to extract variable names of datasets to fill entries in Column.csv file mentioned earlier.
Each row of Column.csv file contains dataset name, variable names separated by space.

```
PROC SQL;
SELECT Name INTO: colum_names SEPARATED BY " " FROM ref WHERE MEMNAME="&&ds_name_&i.";
QUIT;
```

Dataset_Comparison.sas

PROC COMPARE has been used to compare datasets and output dataset of this procedure will be printed on Detailed Difference output file (CSV file) and the comparison result displays in output window (.lst file) is redirected to PDF output file using ODS.

```
PROC COMPARE BASE = &ds1. (obs=&CMP_ROWS)
COMPARE=&ds2. (OBS = &CMP_ROWS) OUT=detailed_diff outnoequal outbase outcomp outdif
MAXPRINT = &No_of_differences_printed.;
%if %UPCASE(&Sort_Required)=YES %then %do;
BY &Sort_Columns1;
%END;
%IF %UPCASE(&Filter_Required)=YES %then %do;
WHERE &Where;
%END;
RUN;
```

Return code of PROC COMPARE has been used to identify exact section of datasets cause mismatches.
In code return code has been converted into 16 digit binary code and there is a different meaning of different position in binary code e.g. Data mismatch, data type mismatch, length mismatch etc. and those were assigned in macro variables result1 – result9 as shown in below code. These details will be printed on Result output file (CSV file).

Compare Me Utility Package

Get_Metadata.sh:

```
#!/bin/ksh
```

```
#########################################################################################################################
#####################################################################
# Please keep correct parameter first parm is datasets location second parm is matadata csv file with its location third is colums csv file with location#
# so it would look like -sysparm <dataset location>~<matadata csv file name with location>~<Colums csv files with location>                    #
#########################################################################################################################
#####################################################################
```

sas get_metadata.sas -noterminal -noprint -sysparm <dataset's path>~<path>/<metadata file name>.csv~<path>/<file name for columns>.csv

Get_Metadata.sas:

```
%MACRO Metadata(ds_path, op_csv, col_csv);

LIBNAME sasdata &ds_path;
PROC CONTENTS DATA=sasdata._ALL_ MEMTYPE=data
OUT= ref(keep=MEMNAME NAME TYPE LENGTH VARNUM LABEL NOBS) NOPRINT ;
RUN;

PROC SORT data=ref ;
by MEMNAME VARNUM;
RUN;

PROC EXPORT DATA= ref
        OUTFILE= &op_csv
        DBMS=CSV REPLACE;
RUN;

PROC SQL ;
SELECT DISTINCT MEMNAME INTO: dataset_name SEPARATED BY " " FROM ref;
CREATE TABLE ref1 as SELECT DISTINCT MEMNAME FROM ref;
QUIT;

%LET i=1;
%DO %UNTIL (%SYSFUNC (COMPRESS ("%SCAN(&dataset_name., &i)")) = "");
%LET ds_name_&i. = %SCAN(&dataset_name., &i);
PROC SQL ;
SELECT Name INTO: colum_names SEPARATED BY " " FROM ref WHERE MEMNAME="&&ds_name_&i.";
QUIT;

DATA ref1;
LENGTH MEMNAME $50. colum_names $12000.;
SET ref1;
IF MEMNAME = "&&ds_name_&i." THEN colum_names = "&colum_names";
dataset_path=&ds_path.;
RUN;

%LET i = %EVAL(&i.+1);
%END;


PROC EXPORT DATA= ref1
        OUTFILE= &col_csv
        DBMS=CSV REPLACE;
RUN;
%MEND;

OPTIONS noquotelenmax;

%LET a = %SCAN(&sysparm.,1,~);
%LET b = %SCAN(&sysparm.,2,~);
%LET c = %SCAN(&sysparm.,3,~);
```

```
%metadata("&a." , "&b." , "&c.");
```

Dataset_Comparison.sh:

```
#!/bin/ksh
##################################################################################################
# Please keep correct parameter that is the location of Input.csv file so parameter would looks like -sysparm
<Location of Input.csv   #
# Input.csv name is hardcoded in program so never change this file name                          #
##################################################################################################

sas Dataset_Comparison.sas -noterminal -noprint -sysparm <Path of input file Input.csv>
```

Dataset_Comparison.sas:

```
OPTIONS noquotelenmax mlogic mprint;

%MACRO compare_data(ip_op_path);

PROC IMPORT OUT= input DATAFILE= "&ip_op_path/Input.csv"
DBMS=CSV REPLACE; GETNAMES=YES; DATAROW=2;
RUN;

DATA _null_;
SET input nobs=noobs;
CALL SYMPUT("obs_cnt", compress(noobs));
RUN;

%PUT No of records in input xls sheet is &obs_cnt;

%DO i=1 %TO &obs_cnt;
DATA _null_;
SET input;
IF _N_ = &i THEN DO;
CALL SYMPUT("CMP_FORMAT",CMP_FORMAT);
CALL SYMPUT("CMP_COLUMN",CMP_COLUMN);
CALL SYMPUT("CMP_ROWS",CMP_ROWS);
CALL SYMPUT("CMP_FILE1",CMP_FILE1);
CALL SYMPUT("CMP_FILE2",CMP_FILE2);
CALL SYMPUT("CMP_FILENM1",CMP_FILENM1);
CALL SYMPUT("CMP_FILENM2",CMP_FILENM2);
CALL SYMPUT("CMP_OUTPUT",CMP_OUTPUT);
CALL SYMPUT("Sort_Required",Sort_Required);
CALL SYMPUT("Sort_Columns1",Sort_Columns1);
CALL SYMPUT("Sort_Columns2",Sort_Columns2);
CALL SYMPUT("Temp_area",Temp_area);
CALL SYMPUT("Filter_Required",Filter_Required);
CALL SYMPUT("Where",Where);
CALL SYMPUT("colum_map_required",colum_map_required);
CALL SYMPUT("Mapping_colums_of_DS2",Mapping_colums_of_DS2);
CALL SYMPUT("No_of_differences_printed",No_of_differences_printed);
END;
RUN;

%LET diff_in_op_dataset=%EVAL(3*&No_of_differences_printed);

LIBNAME lib1 "&CMP_FILE1";
LIBNAME lib2 "&CMP_FILE2";

%LET result1=;
%LET result2=;
%LET result3=;
%LET result4=;
%LET result5=;
%LET result6=;
%LET result7=;
```

```
%LET result8=;
%LET result9=;

%IF %UPCASE(&CMP_FORMAT)=XLS %THEN %DO;
PROC IMPORT OUT= lib1.&CMP_FILENM1.
        DATAFILE= "&CMP_FILE1/&CMP_FILENM1."
        DBMS=EXCEL REPLACE;

RUN;

PROC IMPORT OUT= lib2.&CMP_FILENM2.
        DATAFILE= "&CMP_FILE2/&CMP_FILENM2."
        DBMS=EXCEL REPLACE;

RUN;

%END;

%IF %UPCASE(&CMP_FORMAT)=CSV %THEN %DO;
PROC IMPORT OUT= lib1.&CMP_FILENM1.
        DATAFILE= "&CMP_FILE1/&CMP_FILENM1..csv"
        DBMS=CSV REPLACE; GETNAMES=YES; DATAROW=2;
RUN;

PROC IMPORT OUT= lib2.&CMP_FILENM2.
        DATAFILE= "&CMP_FILE2/&CMP_FILENM2..csv"
        DBMS=CSV REPLACE;
        GETNAMES=YES; DATAROW=2;
RUN;

%END;


%LET ds2 = lib2.&CMP_FILENM2.;

%IF %UPCASE(&Sort_Required)=YES %THEN %DO;

LIBNAME temp "&Temp_area";
%LET temp_ds_1 = %sysfunc(compress(&CMP_FILENM1.))_1;
%LET temp_ds_2 = %sysfunc(compress(&CMP_FILENM2.))_2;

PROC sort DATA=lib1.&CMP_FILENM1. OUT=temp.&temp_ds_1;
BY &Sort_Columns1;
RUN;

PROC sort DATA=lib2.&CMP_FILENM2. OUT=temp.&temp_ds_2;
BY &Sort_Columns2;
RUN;

%LET ds1 = temp.&temp_ds_1;
%LET ds2 = temp.&temp_ds_2;

%END;
ods pdf file="&CMP_OUTPUT.";

%IF %UPCASE(&CMP_ROWS)=ALL %THEN %LET CMP_ROWS=MAX;
%IF %UPCASE(&CMP_COLUMN)=ALL %THEN %DO;

PROC COMPARE BASE=&ds1. (obs=&CMP_ROWS)
COMPARE=&ds2. (obs=&CMP_ROWS) OUT=detailed_diff outnoequal outbase outcomp outdif
maxprint=&No_of_differences_printed.;
%IF %UPCASE(&Sort_Required)=YES %THEN %DO;
BY &Sort_Columns1;
%END;
%IF %UPCASE(&Filter_Required)=YES %THEN %DO;
WHERE &Where;
```

```
%END;
RUN;


%IF &sysinfo=0 %THEN %DO;
%LET result="Matched successfully";
%END;
%ELSE %DO;
%LET comprc=&sysinfo;
DATA _null_;
comprc=&comprc;
comprcbin=put(comprc,binary16.);
CALL SYMPUT("comprcbin",comprcbin);
RUN;
%PUT The Return Code is : &comprcbin;
%IF %SUBSTR(&comprcbin,01,1)=1 %THEN %LET result1=Error: comparison is not DOne check the SAS log.;
%IF %SUBSTR(&comprcbin,03,1)=1 %THEN %LET result2=Conflicting column datatype.;
%IF %SUBSTR(&comprcbin,04,1)=1 %THEN %LET result3=Data not matched.;
%IF %SUBSTR(&comprcbin,05,1)=1 %THEN %LET result4=Column of second dataset is not in first.;
%IF %SUBSTR(&comprcbin,06,1)=1 %THEN %LET result5=Column of first dataset is not in second.;
%IF %SUBSTR(&comprcbin,09,1)=1 %THEN %LET result6=Second dataset has more rows than first.;
%IF %SUBSTR(&comprcbin,10,1)=1 %THEN %LET result7=first dataset has more rows than Second.;
%IF %SUBSTR(&comprcbin,11,1)=1 %THEN %LET result8=column levels are different.;
%IF %SUBSTR(&comprcbin,12,1)=1 %THEN %LET result9=column lengths are different.;
%LET result="&result3. &result1. &result2. &result4. &result5. &result6. &result7. &result8. &result9.";
%END;
%END;
%ELSE %DO;

%IF %UPCASE(&colum_map_required)=YES %THEN %DO;
DATA &ds2.;
SET &ds2.;
rename
%LET i=1;
%DO %until %SCAN(&Mapping_colums_of_DS2,&i) = "" ;
%SCAN(&Mapping_colums_of_DS2,&i) = %SCAN(&CMP_COLUMN, &i.)
%LET i = %EVAL(&i.+1);
%END;
;
RUN;

PROC COMPARE BASE=&ds1. (obs=&CMP_ROWS)
COMPARE=&ds2. (obs=&CMP_ROWS) OUT=detailed_diff outnoequal outbase outcomp outdif
maxprint=&No_of_differences_printed.;
%IF %UPCASE(&Sort_Required)=YES %THEN %DO;
BY &Sort_Columns1;
%END;
VAR &CMP_COLUMN;
%IF %UPCASE(&Filter_Required)=YES %THEN %DO;
WHERE &Where;
%END;
RUN;

%IF &sysinfo=0 %THEN %DO;
%LET result="Matched successfully";
%END;
%ELSE %DO;
%LET comprc=&sysinfo;
DATA _null_;
comprc=&comprc;
comprcbin=put(comprc,binary16.);
CALL SYMPUT("comprcbin",comprcbin);
RUN;
%PUT The Return Code is : &comprcbin;
%IF %SUBSTR(&comprcbin,01,1)=1 %THEN %LET result1=Error: comparison is not done check the SAS log.;
%IF %SUBSTR(&comprcbin,03,1)=1 %THEN %LET result2=Conflicting column datatype.;
```

```
%IF %SUBSTR(&comprcbin,04,1)=1 %THEN %LET result3=Data not matched.;
%IF %SUBSTR(&comprcbin,05,1)=1 %THEN %LET result4=Column of second dataset is not in first.;
%IF %SUBSTR(&comprcbin,06,1)=1 %THEN %LET result5=Column of first dataset is not in second.;
%IF %SUBSTR(&comprcbin,09,1)=1 %THEN %LET result6=Second dataset has more rows than first.;
%IF %SUBSTR(&comprcbin,10,1)=1 %THEN %LET result7=first dataset has more rows than Second.;
%IF %SUBSTR(&comprcbin,11,1)=1 %THEN %LET result8=column levels are different.;
%IF %SUBSTR(&comprcbin,12,1)=1 %THEN %LET result9=column length are different.;
%LET result="&result3. &result1. &result2. &result4. &result5. &result6. &result7. &result8. &result9.";
%END;

%PUT &result.;

%END;
ods pdf close;

DATA temp_&i.;
SET input;
RESULT=&result;
IF _N_ = &i THEN output;
RUN;

%IF &i=&obs_cnt %THEN %DO;
DATA result (Keep=CMP_FILE1 CMP_FILENM1 CMP_FILE2 CMP_FILENM2 RESULT);
SET
%DO j=1 %TO &obs_cnt;
temp_&j.
%END;
;
RUN;

PROC EXPORT DATA=result
        OUTFILE="&ip_op_path/Result.csv"
        DBMS=CSV REPLACE;
RUN;

%END;

DATA detailed_diff; SET detailed_diff (obs=&diff_in_op_dataset.);RUN;

PROC EXPORT DATA=detailed_diff
        OUTFILE="&ip_op_path/detailed_differences_&i..csv"
        DBMS=CSV REPLACE;
RUN;

  PROC DATASETS nolist library = temp;
     delete &temp_ds_1 &temp_ds_2;

  RUN;

%END;
%MEND;

%compare_data(&sysparm.);
```

Input.csv:

Create a CSV file give following column names in first row.
CMP_FORMAT, CMP_COLUMN, CMP_ROWS, CMP_FILE1, CMP_FILE2, CMP_FILENM1, CMP_FILENM2, CMP_OUTPUT, Sort_Required, Sort_Columns1, Sort_Columns2, Temp_area, Filter_Required, Where, colum_map_required, Mapping_colums_of_DS2, No_of_differences_printed

Conclusion

This tool is very easy to use and reduces the effort required to write a code for data comparison every time when comparison of two or more SAS datasets is required. It could be useful during the testing phase of any application migration or application development projects, where there is a need to compare the datasets of a legacy application prior to migration/upgrade against the datsets of the migrated application post migration/upgrade. The tool can be easily configured across different environments such as multiple operating systems and platforms without having to update or enhance the code by a huge measure. This tool can be also be easily used by a testers or developers having limited SAS knowledge.

References

http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000057814.htm

Acknowledgements

We would also like to thank our Cognizant managers Subburaj Krishnasamy, Dhananjay Kelkar & Prakash Pothemsetti for encouraging our efforts to document our research and present a paper. We would also like to thank Neelakandan Vishwanathan who was actively involved in suggesting the requirements for creation of such a tool and was also heavily involved in validating the comparison results.

Contact Information

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

Name : Anurag Katare
Address : 56A Van Wyk Road
City, State ZIP : Lake Hiawatha, 07034
Phone: 862-579-7988
E-mail: er.anuragkatare@gmail.com

Name: Jayesh Soneji
Address: 465 Meadow Rd,
City, State, Zip: Princeton, NJ, 08650
Phone: 201-682-5909
E-mail: jayssoneji@gmail.com