SAS® GLOBAL FORUM 2018

USERS PROGRAM

Comparisons of Large Complex Data Sets with Matching Common Dimensions to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc., Brecksville, Ohio

April 8 – 11 | Denver, CO
#SASGF

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Abstract

Today, with the tremendous size of data files and frequent changes, comparing complex, large data files is a growing challenge. It is impossible to only use PROC COMPARE to catch the differences between current and previous large, complex data files with different numbers of observations and lots of changes.

Fortunately, SAS offers a simple solution. We split the current and previous data into **three parts**:

1) the common dimensions with **modified** values exists in current and previous data files;

2) an additional part **new added** into current data file; and

3) an additional part **dropped** from previous data file.

Then we can compare the common dimensions, catch what was modified; and catch the additional parts in current and previous data.

Key words

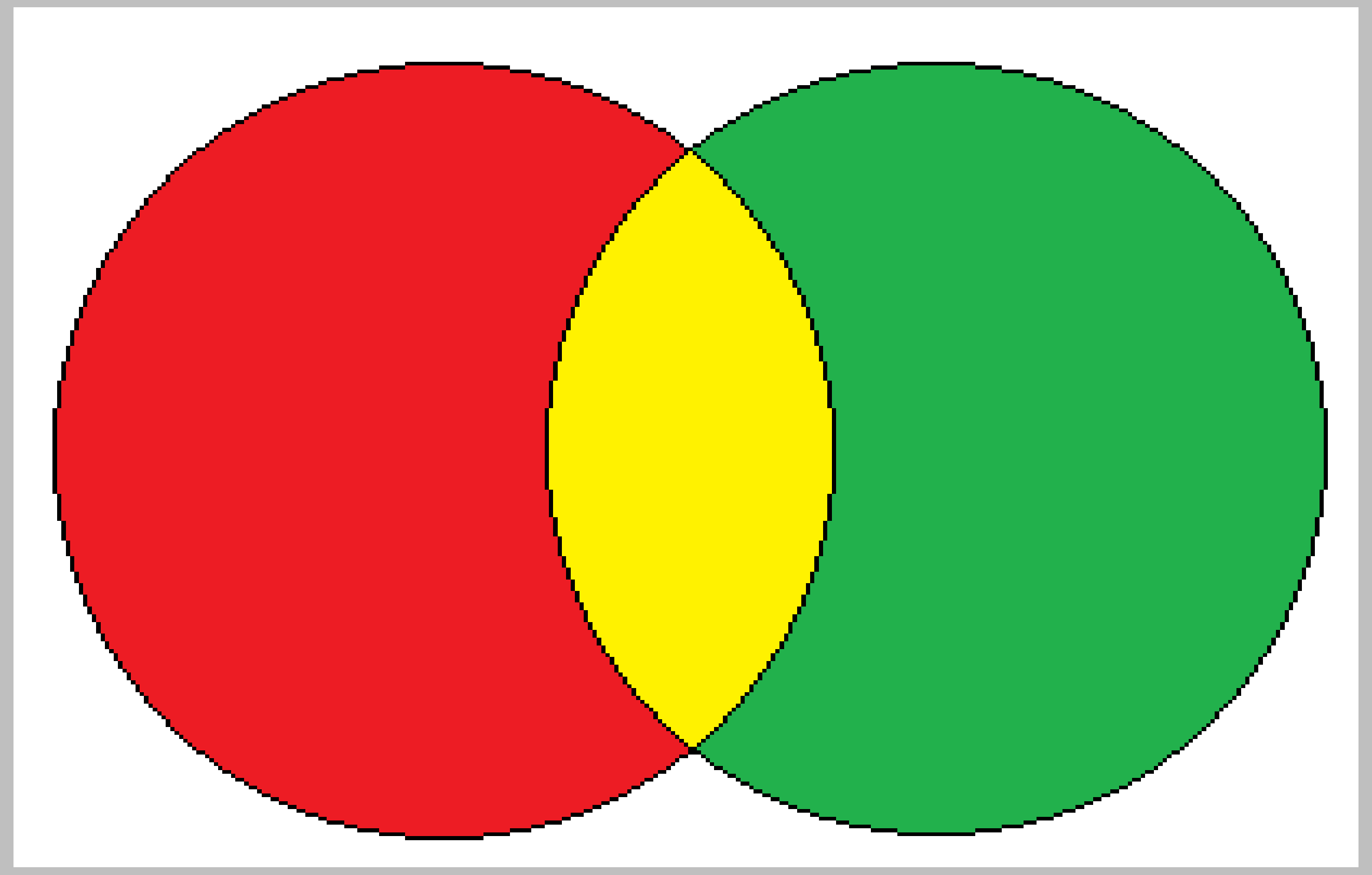
PROC COMPARE,

large and complex data files comparison,

Red: the additional part **new added** into current data;

Green: the additional part **dropped** from previous data;

Yellow: the common dimensions part **modified** in both data files;



Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Warning

Before starting comparison, I suggest using **PROC EXPORT to export the current and previous data files as txt files** if both were with different lengths or formats, then **use DATA STEP INFILE to read them as .sas7bdat files with standardized lengths and formats**.

For example, for the same column variable, it may only include numeric values in one file, but include character and numeric values in another file. Under this situation, if using PROC IMPORT may cause trouble during comparison.

Comparison engine

Users only need **to modify the variables and paths correspondingly** in the very beginning because we use automation technical skills to reduce manual modifications everywhere.

Here is the engine:

```
/*use to show the values of macro and their parameters*/
options mprint symbolgen mlogic;
%let root = /sas/development/reports/complex_files;
%let D_CUR =&root./current;
LIBNAME D_CUR "&D_CUR";        /*current output path */

%let D_PREV=&root./previous;
LIBNAME D_PREV "&D_PREV";     /*previous output path */

%let D_FINAL=& root./current_vs_prev;
LIBNAME D_FINAL "&D_FINAL";    /*final comparison output path */
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Compare engine (cur,prv)

```
%macro data_reader_cleanner(path=,name=);
data &name(drop=Run_Date Release_Number);  /*Run_Date and
Release_Number always different*/
   infile "&path./&name..txt" dlm='|' dsd missover pad firstobs=2
lrecl=1025;
   input  Release_Number :8. Run_Date :$50. X1 :$50. X2 :$50.  X3
:$50.  X4 :$50.  X5 :$50.   X6 :$50. X7 :$50.  X8 :$50. X9 :$100. X10
:$50. X11 :$50.  X12 : $50.   %do i = 0 %to 200; TIME&i  :8.  %end; ;
run;

/***** set missing values to be zeros******/
/*Here we can replace all the following TIME: with _numeric_*/
proc stdize data=&name reponly missing=0 out=&name._0;
   var TIME:;
run;
```

Compare engine (continued)

```
/* delete the row if all columns are zeros */
/* we may also replace all the following TIME: with _numeric_*/
data &name._no0;
      set &name._0;
array var[*] TIME:;        grandtotalsum=0;
   do z=1 to dim(var);     grandtotalsum=grandtotalsum + abs(var[z]);     end;
if grandtotalsum = 0 then delete;
drop  z grandtotalsum;
run;

/*sort by _ALL_ for comparisons, only sorted data sets are good for comparison*/
proc sort data=&name._no0     out=&name._srt ;
by _ALL_ ;
run;
%mend data_reader_cleanner;
%data_reader_cleanner(path=&D_CUR,name=cur);
%data_reader_cleanner(path=&D_PREV,name=prv);
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Compare engine (continued)

```
/***roughly compare first, if exactly equal, then the rest of the files
would be empty **/
proc compare base=cur_srt compare=prv_srt out=cur_prv_diff;
run;

/*set common columns as sort keys, output them as txt file*/
/*The following procedures can also be replaced by proc SQL*/
/*select … into : param_name separated by ‘ ‘ from source data*/
proc export data=cur_prv_diff(obs=0 drop=TIME:  _TYPE_   _OBS_)
outfile="&D_FINAL./used_for_srtd.txt"  dbms=csv replace;
delimiter=" ";          putnames=YES;
run;
```

Compare engine (continued)

```
/*define the txt file with common columns as sort keys*/
filename COLLST "&D_FINAL./";
%macro used_for_srtd_txt;
%include COLLST("used_for_srtd.txt");
%mend used_for_srtd_txt;
/*find common columns and compare them*/
data find_commonkeys;
merge cur_srt(drop=TIME: in=a)
          prv_srt(drop=TIME: in=b);
   by  _ALL_ ;
if a=1 and b=1 ;
run;
 proc sort data=find_commonkeys;
    by %used_for_srtd_txt;
run;
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Compare engine (continued)

```
/*select the common values from current and previous datasets*/
%macro common_values_frm_current_prev(data_name=);
proc sort data=&data_name._srt;
    by %used_for_srtd_txt;
run;
data select_common_values_from_&data_name.1;
merge find_commonkeys(in=a) &data_name._srt(in=b);
  by %used_for_srtd_txt;
if a=1 and b=1 ;
run;
proc sort data=select_common_values_from_&data_name.1
out=select_common_val_from_&data_name._2;
      by %used_for_srtd_txt;
run;
```

Compare engine (continued)

```
/* create OBS_ID as merge key here because will add true
values of character columns back into Comparison results in
the comparison result .sas7bdat file, characters are. if the
letters are same */
data common_var_same_sequence_&data_name;
OBS_ID=_N_;
set select_common_val_from_&data_name._2;
run;
%mend  common_values_frm_current_prev;
%common_values_frm_current_prev(data_name=prv);
%common_values_frm_current_prev(data_name=cur);
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Compare engine (continued)

```
/*that OBS_ID is not equal 0 means the sequence order has issues
or unmatch order sequences avoid to use by statement because it is
very time-consuming */
proc compare base=common_var_same_sequence_cur
             compare=common_var_same_sequence_prv
out=common_part_diff_cur_vs_prv(rename=(_OBS_=OBS_ID_NEW))
criterion=0.00001;
run;

/*if any OBS_ID is not 0, then code will automatically stop
executing*/
data stop_triger_for_unmatch;
set common_part_diff_cur_vs_prv;
if OBS_ID ^=0 then stop;
run;
```

Compare engine (continued)

```
data diff_cur_vs_prv_renamed(rename=(OBS_ID_NEW=OBS_ID));
set stop_triger_for_unmatch(drop=OBS_ID _TYPE_);
run;

/*put common values of character variables back into the
comparison result table*/
data modified_rows(drop=_TYPE_ _OBS_);
merge common_var_same_sequence_cur (drop=TIME: in=a)
      diff_cur_vs_prv_renamed   (keep=OBS_ID TIME: in=b);
   by OBS_ID;
run;
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Compare engine (continued)

```
/*remove the rows if all numeric columns are only 0's, if the whole row numeric values are 0’s, it means that this row is exactly same on both current and previous parts.  we use abs(var[z]) LT 0.00001 then var[z]=0; because system differences always exist*/
data D_FINAL.common_modified_rows;
set modified_rows(drop=OBS_ID);
array var[*] TIME:;
grandtotalsum=0;
do z=1 to dim(var);
if abs(var[z]) LT 0.00001 then var[z]=0;
grandtotalsum=grandtotalsum + abs(var[z]);
end;
if grandtotalsum = 0 then delete;
drop  z grandtotalsum;
run;
```

Compare engine (continued)

```
 /*find additionl values from previous and current data sets*/
%macro additnl_values_frm_current_prev(file_name=);
data D_FINAL.additnl_rows_in_&file_name;
 merge find_commonkeys(in=a)
           &file_name._srt(in=b);
       by %used_for_srtd_txt;
       if a^=1 and b=1;
run;
%mend additnl_values_frm_current_prev;
%additnl_values_frm_current_prev(file_name=prv);
%additnl_values_frm_current_prev(file_name=cur);
```

Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

Conclusion

In the D_FINAL folder, if additnl_rows_in_prv, additnl_rows_in_cur and common_modified_rows are **empty**, it means that there are no differences and no additional or deleted rows in the large, complex data sets.

Bio information

I am a Sr. SAS Tech Lead, SR. SAS Developer and Sr. predictive modeler with 9 year experience in SAS programming, 3 year in developing SAS engines.

My major was History, I got my master degree in History from East China Normal University; got master degree of applied mathematics from University of New Orleans; master degree of Statistics from University of Wyoming. I am expert at developing, optimizing SAS engines.

My main achievements include: 1) Reducing at least 80%-90% of the execution time of model engines using mixed methods; 2) Using automation technical skills such as the SAS Code Automatic Generation Method to fulfill automation run of our model engines. 3) Developed more than 20 automatic execution banking models engines in our bank; 4) fitted some predictive modeling, developed many very good forecasting models using statistical modeling and machine learning methods.

My motto is that the beauty of our engines developing should be: using the simplest codes to solve difficult and complex tasks, to develop SAS engines that can automatically cover all expected changes, fulfill automation run, and finish engines execution in the shortest time.

I am also a part-time media reporter and freelancer.

- Comparisons of Large Complex Data Sets to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc.

CONTACT INFORMATION

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

Kaiqing Fan
Mastech Digital Inc.
Address: 6750 Miller Road, Brecksville, OH-44141
Mobile: 504.344.7267
Email: fankaiqinguw@gmail.com
Linkedin: https://www.linkedin.com/in/fan-kaiqing-81776940/

SAS® GLOBAL FORUM 2018

April 8 – 11 | Denver, CO
Colorado Convention Center

#SASGF

Comparisons of Large Complex Data Sets with Matching Common Dimensions to catch any Changes using SAS® Enterprise Guide

Kaiqing Fan, Mastech Digital Inc., Brecksville, Ohio

Abstract

Today, with the tremendous size of data files and frequent changes, comparing complex, large data files is a growing challenge. It is impossible to only use PROC COMPARE to catch the differences between current and previous large, complex data files with different numbers of observations and lots of changes. Fortunately, SAS offers a simple solution. We split the current and previous data into three parts: 1) the common dimensions with **modified** values in current and previous data files; 2) an additional part **new added** into current data file; and 3) an additional part **dropped** from previous data file. Then we can compare the common dimensions, catch what was modified; and catch the additional parts in current and previous data.

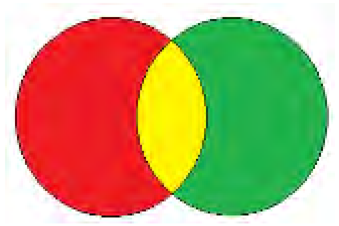
Key words: PROC COMPARE, large and complex data files, the common dimensions with modified, the additional part in previous file, the additional part in current file

Introduction

In banking, the tremendously large data files, their volumes, variables and values, and the number of observations are all continuously changing; business requirements keep changing, too. It's a great challenge to compare the results of these data files from different executions. What we always want to know is the difference between the current and previous runs.

If we only use PROC COMPARE, it is impossible to catch the differences between the current and previous version data sets because the current data and previous data may have different number of observations, many different values.

The solution is to split the current data and previous data into three parts as shown in the following colorful image: 1) the common dimensions parts with modified values in which exists in both current and previous data files; 2) the additional part only exists in current data file; and 3) the additional part only exists in previous data file. Then we can compare with common dimensions, catch what are modified; and catch the additional parts only in both current data and previous data.



Red: the additional part **new added** into current data;

Green: the additional part **dropped** from previous data;

Yellow: the common dimensions part **modified** in both data files;

Comparison Engine

Before starting comparison, I suggest using PROC EXPORT to export the current and previous data files as txt files if both were with different lengths or formats, then use DATA STEP INFILE to read them as .sas7bdat files with standardized lengths and formats. For example, for the same column variable, it may only include numeric values in one file, but include character and numeric values in another file, under this situation, if using PROC IMPORT may cause trouble during comparison.

Here is the engine. Users only need to modify the variables and paths correspondingly in the very beginning. In this engine, I tried to use automation technical skills to reduce manual modifications. I will also list some other automation skills if possible.

```
/*use to show the values of macro and their parameters*/
options mprint symbolgen mlogic;
%let root = /sas/development/reports/ complex_files;
%let D_CUR =&root. /current;
LIBNAME D_CUR "&D_CUR";        /*current output path */

%let D_PREV=&root. /previous;
LIBNAME D_PREV "&D_PREV";     /*previous output path */

%let D_FINAL=& root./current_vs_prev;
LIBNAME D_FINAL "&D_FINAL";   /*final comparison output path */

/**************************************************/
/*******            compare_engine(cur,prv);               ****/
```

```
/*****************************************************/
%macro data_reader_cleanner(path=,name=);
data &name(drop=Run_Date Release_Number);  /*Run_Date and Release_Number always different*/
   infile "&path./&name..txt" dlm='|' dsd missover pad firstobs=2 lrecl=1025;
   input  Release_Number :8. Run_Date :$50. X1 :$50. X2 :$50.  X3 :$50.  X4 :$50.  X5 :$50.
   X6 :$50. X7 :$50.  X8 :$50. X9 :$100. X10 :$50. X11 :$50.  X12 : $50.
   %do i = 0 %to 200; TIME&i  :8.  %end; ;
run;

/***** set missing values to be zeros******/
/*Here we can replace all the following TIME: with _numeric_*/
proc stdize data=&name reponly missing=0 out=&name._0;
   var TIME:;
run;

/* delete the row if all columns are zeros */
/* we may also replace all the following TIME: with _numeric_*/
data &name._no0;
      set &name._0;
           array var[*] TIME:;
           grandtotalsum=0;
              do z=1 to dim(var);
                  grandtotalsum=grandtotalsum + abs(var[z]);
              end;
           if grandtotalsum = 0 then delete;
           drop  z grandtotalsum;
run;

/*sort by _ALL_ for comparisons, only sorted data sets are good for comparison*/
proc sort data=&name._no0     out=&name._srt ;
           by _ALL_ ;
run;
%mend data_reader_cleanner;
options mprint symbolgen mlogic;
%data_reader_cleanner(path=&D_CUR,name=cur);
%data_reader_cleanner(path=&D_PREV,name=prv);

/***roughly compare first, if exactly equal, then the rest of the files would be empty **/
proc compare base=cur_srt compare=prv_srt out=cur_prv_diff;   run;

/*find common columns set them as sort keys, output them as txt file*/
/*The following two procedures can also be replaced by proc SQL;*/
/* select … into : parameter_name separated by ‘ ‘ from source data */
proc export data=cur_prv_diff(obs=0 drop=TIME:  _TYPE_  _OBS_)
           outfile="&D_FINAL./used_for_srtd.txt"  dbms=csv replace;
           delimiter=" ";
            putnames=YES;
run;
```

```
/*define the txt file inside with common columns as sort keys using macro*/
filename COLLST "&D_FINAL./";
%macro used_for_srtd_txt;
%include COLLST("used_for_srtd.txt");
%mend used_for_srtd_txt;

/*find common columns and compare them*/
data find_commonkeys;
          merge cur_srt(drop=TIME: in=a)
                prv_srt(drop=TIME: in=b);
   by _ALL_ ;
if a=1 and b=1 ;
run;

proc sort data=find_commonkeys;
   by %used_for_srtd_txt;
run;

/*select the common values from current and previous datasets*/
%macro common_values_frm_current_prev(data_name=);
proc sort data=&data_name._srt;
   by %used_for_srtd_txt;
run;

data select_common_values_from_&data_name.1;
          merge find_commonkeys(in=a) &data_name._srt(in=b);
  by %used_for_srtd_txt;
if a=1 and b=1 ;
run;

proc sort data=select_common_values_from_&data_name.1
          out=select_common_val_from_&data_name._2;
    by %used_for_srtd_txt;
run;

/* create OBS_ID as merge key here because will add true values of character columns back into
Comparison results in the comparison result .sas7bdat file, characters are. if the letters are same */
data common_var_same_sequence_&data_name;
          OBS_ID=_N_;
          set select_common_val_from_&data_name._2;
run;
%mend  common_values_frm_current_prev;
options mprint symbolgen mlogic;
%common_values_frm_current_prev(data_name=prv);
%common_values_frm_current_prev(data_name=cur);

/*that OBS_ID is not equal 0 means the sequence order has issues or unmatch order sequences*/
```

```
/*avoid to use by statement because it is very time-consuming */
proc compare base=common_var_same_sequence_cur
                    compare=common_var_same_sequence_prv
        out=common_part_diff_cur_vs_prv(rename=(_OBS_=OBS_ID_NEW)) criterion=0.00001;
run;

/*if any OBS_ID is not 0, then code will automatically stop executing*/
data stop_triger_for_unmatch;
            set common_part_diff_cur_vs_prv;
            if OBS_ID ^=0 then stop;
run;

data diff_cur_vs_prv_renamed(rename=(OBS_ID_NEW=OBS_ID));
            set stop_triger_for_unmatch(drop=OBS_ID _TYPE_);
run;

/*put common values of character variables back into the comparison result table*/
data modified_rows(drop=_TYPE_ _OBS_);
            merge common_var_same_sequence_cur (drop=TIME: in=a)
                     diff_cur_vs_prv_renamed   (keep=OBS_ID TIME: in=b);
      by OBS_ID;
run;

/*remove the rows if all numeric columns are only 0's*/
/*if the whole row numeric values are 0's, */
/*it means that this row is exactly same on both current and previous parts*/
/* we use abs(var[z]) LT 0.00001 then var[z]=0; because system differences always exist*/
data D_FINAL.common_modified_rows;
            set modified_rows(drop=OBS_ID);
                array var[*] TIME:;
                grandtotalsum=0;
                      do z=1 to dim(var);
                                  if abs(var[z]) LT 0.00001 then var[z]=0;
                                  grandtotalsum=grandtotalsum + abs(var[z]);
                      end;
               if grandtotalsum = 0 then delete;
              drop  z grandtotalsum;
run;

/*find additionl values from previous and current data sets*/
%macro additnl_values_frm_current_prev(file_name=);
data D_FINAL.additnl_rows_in_&file_name;
            merge find_commonkeys(in=a)
                     &file_name._srt(in=b);
      by %used_for_srtd_txt;
      if a^=1 and b=1;
run;
%mend additnl_values_frm_current_prev;
```

```
options mprint symbolgen mlogic;
%additnl_values_frm_current_prev(file_name=prv);
%additnl_values_frm_current_prev(file_name=cur);
```

Conclusion

In the D_FINAL folder, if additnl_rows_in_prv, additnl_rows_in_cur and common_modified_rows are empty, it means that there are no differences and no additional or deleted rows in the large, complex data sets.

ACKNOWLEDGMENTS

I would like to thank AnnMaria De Mars very much for reviewing this paper, and mentoring me a lot!

CONTACT INFORMATION

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

Kaiqing Fan
Mastech Digital Inc.
Address: 6750 Miller Road, Brecksville, OH-44141
Mobile: 504.344.7267
Email: fankaiqinguw@gmail.com
Linkedin: https://www.linkedin.com/in/fan-kaiqing-81776940/