MWSUG 2018-Paper111

How to Avoid Possible Tricks

When Using DATA STEP MERGE Instead of PROC SQL JOIN

Guangtao Gao
Master Student of Applied Statistics
Cleveland State University

ABSTRACT

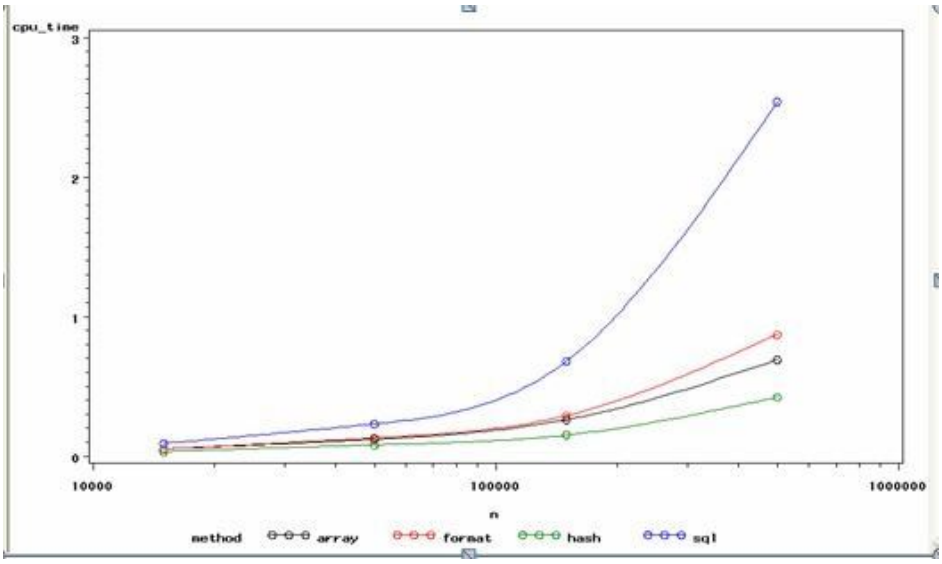
PROC SQL JOIN can avoid lots of potent troubles, but costs too much execution time; DATA STEP MERGE can run much faster, however it has some potential tricks, avoiding these tricks carefully, DATA STEP MERGE would be a better choice. How to avoid these potential tricks would be the following topic. I also correct one popular beautiful method to modify the different lengths of variables directly using length statement.

INTRODUCTION

Why I don't like PROC SQL JOIN

When we merge or join large data files, PROC SQL JOIN can avoid lots of potent troubles, but it costs too much execution time; DATA STEP MERGE can run much faster, however it has some potential tricks. If we avoid these tricks carefully, DATA STEP MERGE would be a better choice.

An article named as *Performance Analysis of different equivalents of Proc SQL in SAS* shows that PROC SQL is not the perfect choice. Hash Object and DATA STEP MERGE are the two best options. Below is the decreasing order of performance time when joining datasets in SAS.



F1.Hash object Merge vs Data step vs Proc SQL

Many SAS developers discussed that using DATA STEP MERGE may cause lots of potential risks such as truncate the end letters when merging without warnings. Here I want to share my experience about avoid the possible tricks when using DATA STEP MERGE.

Tips to avoid the potential risks when using DATA STEP MERGE

1) Before merging, standardize their sort keys with same lengths or formats

As a SAS developer, we may often see the following WARNING:

WARNING: Multiple lengths were specified for the BY variable **Name** by input data sets. This might cause unexpected results.

It is important to handle multiple lengths. I use the dataset-CLASS from SASHELP library. In kids_class data file, Name is a character variable with 11 lengths, Gender is a character variable with 6 lengths.

```
data kids_class(rename=(Name_new =Name));
retain Name_new  Gender;
set SASHELP.class;
length Gender $6. Name_new $ 11.;
if sex ='M' then Gender='Male';
else            Gender='Female';
Name_new = left(strip(Name));
drop Name;
run;
```

Way1, using length statement to change the lengths of sorted keys may have potential risks.

Actually this way is very popularly suggested online and papers by some SAS programmers. If we use PROC IMPORT to read in xlsx or csv files, we cannot use **length statement** such as **length Name $ 11. Gender $ 6.** simply, because it will cause truncation.

```
%let path = /sas/model/ reports/models/combination_sas/test_merge;
proc import out=boy_class_xlsx datafile="&path./boy_class.xlsx" dbms=xlsx replace;
getnames=YES;                    /*here the length of Name is $ 10. , Gender is $ 4. */
run;
OR
/*proc import csv file*/
```

```
proc import out=boy_class_csv datafile="&path./kids_class.csv" dbms=csv replace;
delimiter=' ';  getnames=YES;      /*here the length of Name is $ 10. , Gender is $ 4. */
run;
/*bad codes*/
/*reset the length of the sorted keys here does not truly change their lengths*/
data boy_class_xlsx_relength;
length Name $ 11. Gender $ 6.;
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_relength out=boy_class_xlsx_relength_srt;
by Name Gender;
run;
proc sort data=kids_class                  out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name Gender;
run;
data merged_class_xlsx_relength;
/*length Name $ 11. Gender $ 6.;  */
merge boy_class_xlsx_relength_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;
```

BOY_CLASS_XLSX_RELENGTH_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyz

	Name	Gender	Sex	Age
1	Alfred	MOle	M	17
2	Henry	MPle	M	14
3	James	MAle	M	12
4	Jeffrey	Male	M	19
5	KaiqingFan	Male	M	14
6	Robert	Male	M	12
7	Thomas	Male	M	31
8	William	Male	M	15

KIDS_CLASS_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Alice	Female	F	13	56.5	84
3	Barbara	Female	F	13	65.3	98
4	Carol	Female	F	14	62.8	102.5
5	Henry	Male	M	14	63.5	102.5
6	James	Male	M	12	57.3	83
7	Jane	Female	F	12	59.8	84.5
8	Janet	Female	F	15	62.5	112.5
9	Jeffrey	Male	M	13	62.5	84
10	John	Male	M	12	59	99.5
11	Joyce	Female	F	11	51.3	50.5
12	Judy	Female	F	14	64.3	90
13	Louise	Female	F	12	56.3	77
14	Mary	Female	F	15	66.5	112
15	Philip	Male	M	16	72	150
16	Robert	Male	M	12	64.8	128
17	Ronald	Male	M	15	67	133
18	Thomas	Male	M	11	57.5	85
19	William	Male	M	15	66.5	112

MERGED_CLASS_XLSX_RELENGTH

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Alice	Fema	F	13	56.5	84
3	Barbara	Fema	F	13	65.3	98
4	Carol	Fema	F	14	62.8	102.5
5	Henry	Male	M	14	63.5	102.5
6	James	Male	M	12	57.3	83
7	Jane	Fema	F	12	59.8	84.5
8	Janet	Fema	F	15	62.5	112.5
9	John	Male	M	12	59	99.5
10	Joyce	Fema	F	11	51.3	50.5
11	Judy	Fema	F	14	64.3	90
12	Louise	Fema	F	12	56.3	77
13	Mary	Fema	F	15	66.5	112
14	Philip	Male	M	16	72	150
15	Ronald	Male	M	15	67	133

We can avoid the above truncation by switching positions of the two tables:

```
data merged_class_xlsx_relength;

/*length Name $ 11. Gender $ 6.;  */

Merge kids_class_srt(in=a)  boy_class_xlsx_relength_srt(in=b);
```

```
by Name Gender;

if a^=1 and b=1;

run;
```

Way2, creating new variables with standard lengths using old sorted keys.

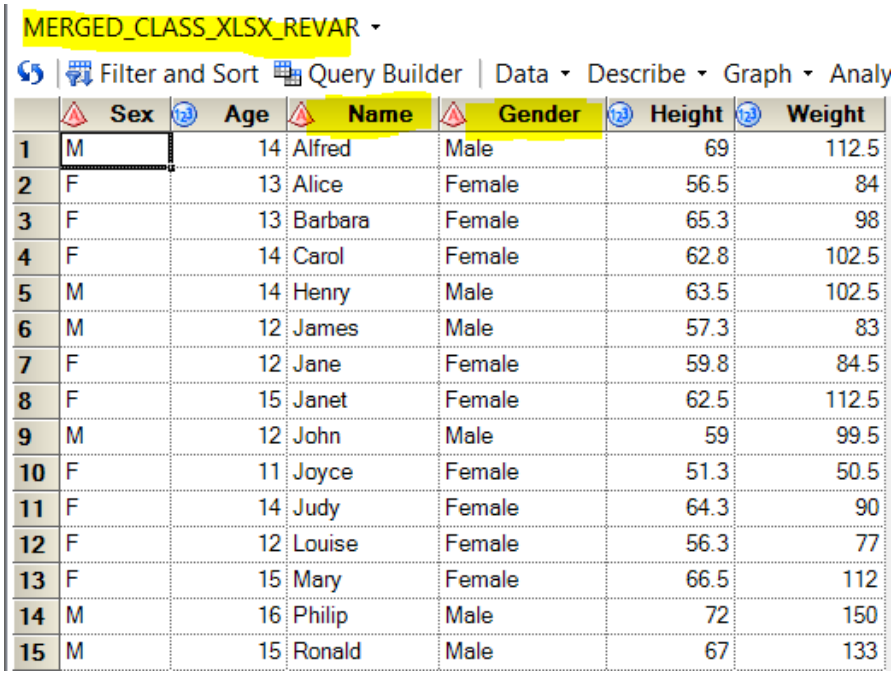
To create new variables with standard lengths from old sorted keys is a good way to standardize multiple lengths and avoid truncations.

```
/*good codes using new variables creation method*/
/*to create different variables with standard lengths for the sorted keys here does truly standardize their lengths*/
data boy_class_xlsx_revar(rename=(Name_new=Name Gender_new=Gender));
set boy_class_xlsx;
length Name_new $ 11. Gender_new $ 6.;
Name_new = left(strip(Name));
Gender_new = strip(Gender);
drop Name Gender;
run;
proc sort data=boy_class_xlsx_revar out=boy_class_xlsx_revar_srt;
by Name Gender;
run;
proc sort data=kids_class              out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name Gender;
run;

data merged_class_xlsx_revar;

merge boy_class_xlsx_revar_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
```

```
run;
```



MERGED_CLASS_XLSX_REVAR

Filter and Sort | Query Builder | Data | Describe | Graph | Analy

	Sex	Age	Name	Gender	Height	Weight
1	M	14	Alfred	Male	69	112.5
2	F	13	Alice	Female	56.5	84
3	F	13	Barbara	Female	65.3	98
4	F	14	Carol	Female	62.8	102.5
5	M	14	Henry	Male	63.5	102.5
6	M	12	James	Male	57.3	83
7	F	12	Jane	Female	59.8	84.5
8	F	15	Janet	Female	62.5	112.5
9	M	12	John	Male	59	99.5
10	F	11	Joyce	Female	51.3	50.5
11	F	14	Judy	Female	64.3	90
12	F	12	Louise	Female	56.3	77
13	F	15	Mary	Female	66.5	112
14	M	16	Philip	Male	72	150
15	M	15	Ronald	Male	67	133

Way3, using format statement to standardize the sorted keys' lengths or formats.

Format statement with format Name $ 11. Gender $ 6. is the best way to fix multiple lengths and avoid truncations.

```
/*good codes using format method*/
data boy_class_xlsx_format;
format Name $ 11. Gender $ 6.;          /*reformat the sorted keys here*/
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_format out=boy_class_xlsx_format_srt;
by Name Gender;
run;
proc sort data=kids_class                out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name Gender;
```

```
run;
data merged_class_xlsx_format;
/*format Name $ 11. Gender $ 6.; */   /* or reformat the sorted keys here*/
merge kids_class_srt(in=a)    boy_class_xlsx_format_srt(in=b);
by Name Gender;
if a=1 and b=1;
run;
```

MERGED_CLASS_XLSX_FORMAT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Alice	Female	F	13	56.5	84
3	Barbara	Female	F	13	65.3	98
4	Carol	Female	F	14	62.8	102.5
5	Henry	Male	M	14	63.5	102.5
6	James	Male	M	12	57.3	83
7	Jane	Female	F	12	59.8	84.5
8	Janet	Female	F	15	62.5	112.5
9	John	Male	M	12	59	99.5
10	Joyce	Female	F	11	51.3	50.5
11	Judy	Female	F	14	64.3	90
12	Louise	Female	F	12	56.3	77
13	Mary	Female	F	15	66.5	112
14	Philip	Male	M	16	72	150
15	Ronald	Male	M	15	67	133

Compared with the above three methods, format is the simplest one, new variables creation is a good choice, and resetting the lengths using length statement is not recommended.

2) Before merging, if we have to keep other same columns in both tables, standardize the lengths of same columns, the methods are same as in 1).

If the same columns such as Gender and Sex with the sorted key Name in the following codes in both tables have different lengths, SAS log will show WARNINGS.

```
data boy_class_txt;
input Name $ 11.  Gender $ 7.  Sex $ 2. Age 8.;
datalines;
Alfred     MPle   M 14
Robert     Male   M 12
Thomas     MTle   M 11
William    Male   M 15
KaiqingFan MFle   M 12
run;
proc sort data=boy_class_txt out=boy_class_txt_revar_srt;
by Name;
run;
proc sort data=kids_class      out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name;
run;
data merged_class_txt_revar;
merge kids_class_srt(in=a)  boy_class_txt_revar_srt(in=b);
by Name;
if a=1 and b=1;
run;
```

WARNING: Multiple lengths were specified for the variable Gender by input data set(s). This can cause truncation of data.

WARNING: Multiple lengths were specified for the variable Sex by input data set(s). This can cause truncation of data.

For the common columns in both tables, we'd better not use length statement to standardize their lengths. To create new variables with standard lengths and to use format statement are good options as in 1).

There is a bad example:

```
/*bad codes to cause truncation*/
/*reset the length of the sorted keys here does not truly change their lengths*/
data boy_class_xlsx_relength;
length Name $ 11. Gender $ 6.;
set boy_class_xlsx;    /*this data is from PROC IMPORT XLSX file*/
run;
proc sort data=boy_class_xlsx_relength out=boy_class_xlsx_relength_srt;
 by Name;
run;
proc sort data=kids_class                out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name;
run;
data merged_class_xlsx_relength;
/*length Name $ 11. Gender $ 6.;  */
merge boy_class_xlsx_relength_srt(in=a)   kids_class_srt(in=b);
by Name;
if a^=1 and b=1;
run;
```

BOY_CLASS_XLSX_RELENGTH_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyz

	Name	Gender	Sex	Age
1	Alfred	MOle	M	17
2	Henry	MPle	M	14
3	James	MAle	M	12
4	Jeffrey	Male	M	19
5	KaiqingFan	Male	M	14
6	Robert	Male	M	12
7	Thomas	Male	M	31
8	William	Male	M	15

KIDS_CLASS_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Alice	Female	F	13	56.5	84
3	Barbara	Female	F	13	65.3	98
4	Carol	Female	F	14	62.8	102.5
5	Henry	Male	M	14	63.5	102.5
6	James	Male	M	12	57.3	83
7	Jane	Female	F	12	59.8	84.5
8	Janet	Female	F	15	62.5	112.5
9	Jeffrey	Male	M	13	62.5	84
10	John	Male	M	12	59	99.5
11	Joyce	Female	F	11	51.3	50.5
12	Judy	Female	F	14	64.3	90
13	Louise	Female	F	12	56.3	77
14	Mary	Female	F	15	66.5	112
15	Philip	Male	M	16	72	150
16	Robert	Male	M	12	64.8	128
17	Ronald	Male	M	15	67	133
18	Thomas	Male	M	11	57.5	85
19	William	Male	M	15	66.5	112

MERGED_CLASS_XLSX_RELENGTH

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Name	Gender	Sex	Age	Height	Weight
1	Alice	Fema	F	13	56.5	84
2	Barbara	Fema	F	13	65.3	98
3	Carol	Fema	F	14	62.8	102.5
4	Jane	Fema	F	12	59.8	84.5
5	Janet	Fema	F	15	62.5	112.5
6	John	Male	M	12	59	99.5
7	Joyce	Fema	F	11	51.3	50.5
8	Judy	Fema	F	14	64.3	90
9	Louise	Fema	F	12	56.3	77
10	Mary	Fema	F	15	66.5	112
11	Philip	Male	M	16	72	150
12	Ronald	Male	M	15	67	133

3) Before merging, avoid same column in both tables.

Let's see the following example of keeping same column in both tables.

```
/*good codes using format method*/
data boy_class_xlsx_format;
format Name $ 11. Gender $ 6.;          /*reformat the sorted key Name and common column Gender here*/
```

```
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_format out=boy_class_xlsx_format_srt;
by Name;
run;
proc sort data=kids_class                out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name;
run;
data merged_class_xlsx_format;
/*format Name $ 11. Gender $ 6.; */    /* or reformat the sorted keys here*/
merge kids_class_srt(in=a)    boy_class_xlsx_format_srt(in=b);
by Name;
if a=1 and b=1;
run;
```

KIDS_CLASS_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Ex

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Alice	Female	F	13	56.5	84
3	Barbara	Female	F	13	65.3	98
4	Carol	Female	F	14	62.8	102.5
5	Henry	Male	M	14	63.5	102.5
6	James	Male	M	12	57.3	83
7	Jane	Female	F	12	59.8	84.5
8	Janet	Female	F	15	62.5	112.5
9	Jeffrey	Male	M	13	62.5	84
10	John	Male	M	12	59	99.5
11	Joyce	Female	F	11	51.3	50.5
12	Judy	Female	F	14	64.3	90
13	Louise	Female	F	12	56.3	77
14	Mary	Female	F	15	66.5	112
15	Philip	Male	M	16	72	150
16	Robert	Male	M	12	64.8	128
17	Ronald	Male	M	15	67	133
18	Thomas	Male	M	11	57.5	85
19	William	Male	M	15	66.5	112

BOY_CLASS_XLSX_FORMAT_SRT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyz

	Name	Gender	Sex	Age
1	Alfred	MOle	M	17
2	Henry	MPle	M	14
3	James	MAle	M	12
4	Jeffrey	Male	M	19
5	KaiqingFan	Male	M	14
6	Robert	Male	M	12
7	Thomas	Male	M	31
8	William	Male	M	15

MERGED_CLASS_XLSX_FORMAT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	MOle	M	17	69	112.5
2	Henry	MPle	M	14	63.5	102.5
3	James	MAle	M	12	57.3	83
4	Jeffrey	Male	M	19	62.5	84
5	Robert	Male	M	12	64.8	128
6	Thomas	Male	M	31	57.5	85
7	William	Male	M	15	66.5	112

The values of Gender from b will overwrite the values of Gender from a.

How to avoid this situation? We should be careful to decide which one should be kept, which one should be dropped. In this case, since variable Gender in table

boy_class_xlsx_format_srt has lots of spelling errors, it should be dropped. For a safe and smart choice, we would reasonably drop Sex and Age, only keep Name.

The following example verifies that:

```
proc sort data=boy_class_xlsx_format(keep=Name) out=boy_class_xlsx_format_srt;
by Name;
run;
proc sort data=kids_class   out=kids_class_srt;   /*in kids_class_srt data, Gender $ 6.  Sex $ 1. */
by Name;
run;
data merged_class_xlsx_format;
merge boy_class_xlsx_format_srt(in=a)     kids_class_srt(in=b);
```

```
by Name;
if a=1 and b=1;
run;
```

MERGED_CLASS_XLSX_FORMAT

Filter and Sort | Query Builder | Data | Describe | Graph | Analyz

	Name	Gender	Sex	Age	Height	Weight
1	Alfred	Male	M	14	69	112.5
2	Henry	Male	M	14	63.5	102.5
3	James	Male	M	12	57.3	83
4	Jeffrey	Male	M	13	62.5	84
5	Robert	Male	M	12	64.8	128
6	Thomas	Male	M	11	57.5	85
7	William	Male	M	15	66.5	112

CONCLUSIONS

1. Before merging, if we don't standardize their lengths, we will get troubles.

2. If data files are small, PROC SQL JOIN is preferable; DATA STEP MERGE is better for large data files.

3. Carefully standardize the lengths of sorted keys using format statement and new variables creating methods, remember to validate the results after merging.

4. Except for the sorted keys, we'd better avoid same columns in both tables; if have to keep common columns in both data files, we must standardize their lengths using format statement and new variables creating methods.

REFERENCES

1. Avoiding data set merging problems when by-variable has different lengths

   https://communities.sas.com/t5/Base-SAS-Programming/Avoiding-data-set-merging-problems-when-by-variable-has/td-p/48568

2. Are there any performance benchmarks on PROC SQL (SAS) vs SQL?

   https://www.quora.com/Are-there-any-performance-benchmarks-on-PROC-SQL-SAS-vs-SQL

3. Changing the length of a character variable

   https://communities.sas.com/t5/Base-SAS-Programming/Changing-the-length-of-a-character variable/td-p/15905?nobounce

4. To change the length of an existing variable in a SAS dataset

   https://www.popcenter.umd.edu/resources/research-tools/stats-support/sas-frequently-used-code-revised/to-change-the-length-of-an-existing-variable-in-a-sas-dataset

5. Lengths and formats: the long and short of it
   http://blogs.sas.com/content/sasdummy/2007/11/20/lengths-and-formats-the-long-and-short-of-it/

6. Different types of merging using Data Step or Proc SQL in SAS

   http://mehtahemal.com/sas/different-types-of-merging-using-data-step-or-proc-sql-in-sas/450/

CONTACT INFORMATION

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

Guangtao Gao
Cleveland State University
216-577-7661
ggtaogao@gmail.com

SAMPLE CODES

```
data kids_class(rename=(Name_new =Name));

retain Name_new  Gender;

set SASHELP.class;

length Gender $6. Name_new $ 11.;

if sex ='M' then Gender='Male';

else                Gender='Female';

Name_new = left(strip(Name));

drop Name;

run;


/*proc import xlsx file*/

%let path = /sas/model/model_dev/enterprise_reports/models/aggregation_sas/test_merge;

proc import out=boy_class_xlsx datafile="&path./boy_class.xlsx" dbms=xlsx replace;

getnames=YES;  run;
```

```
proc contents data=boy_class_xlsx;  run;

proc contents data=kids_class;          run;


/*using different lengths to merge codes*/

proc sort data=boy_class_xlsx out=boy_class_xlsx_diff_length_srt;    by Name Gender;    run;

proc sort data=kids_class out=kids_class_srt;                                    by Name Gender; run;

data merged_class_table_a_b1;

merge kids_class_srt(in=a)   boy_class_xlsx_diff_length_srt(in=b);

by Name Gender;

if a^=1 and b=1;

run;


data merged_class_table_a1_b;

merge kids_class_srt(in=a)   boy_class_xlsx_diff_length_srt(in=b);

by Name Gender;

if a=1 and b^=1;

run;


data merged_class_table_kids_boy;

merge kids_class_srt(in=a)   boy_class_xlsx_diff_length_srt(in=b);
```

```
by Name Gender;
if a^=1 and b=1;
run;

data merged_class_table_b_a;
merge boy_class_xlsx_diff_length_srt(in=b)    kids_class_srt(in=a);
by Name Gender;
if b^=1 and a=1;
run;

/*good codes*/
data boy_class_xlsx_revar(rename=(Name_new=Name Gender_new=Gender));
set boy_class_xlsx;
length Name_new $ 11. Gender_new $ 6.;
Name_new = left(strip(Name));
Gender_new = strip(Gender);
drop Name Gender;
run;
proc sort data=boy_class_xlsx_revar out=boy_class_xlsx_revar_srt;    by Name Gender;    run;
proc sort data=kids_class out=kids_class_srt;   by Name Gender; run;
```

```
data merged_class_xlsx_revar;
merge boy_class_xlsx_revar_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;


/*bad codes*/
data boy_class_xlsx_relength;
length Name $ 11. Gender $ 6.;   /*reset the length of the sorted keys*/
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_relength out=boy_class_xlsx_relength_srt;    by Name Gender;
run;
proc sort data=kids_class out=kids_class_srt;   by Name Gender; run;
data merged_class_xlsx_relength;
/*length Name $ 11. Gender $ 6.;  */  /*reset the length of the sorted keys*/
merge boy_class_xlsx_relength_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;
```

```
data merged_class_xlsx_relength;
merge kids_class_srt(in=a)   boy_class_xlsx_relength_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;

/*good codes*/
data boy_class_xlsx_format1;
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_format1 out=boy_class_xlsx_format1_srt;  by Name Gender;
run;
proc sort data=kids_class out=kids_class_srt;   by Name Gender; run;
data merged_class_xlsx_format1;
format Name $ 11. Gender $ 6.;
merge boy_class_xlsx_format1_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;
```

```
/*good codes*/
data boy_class_xlsx_format;
format Name $ 11. Gender $ 6.;
set boy_class_xlsx;
run;
proc sort data=boy_class_xlsx_format out=boy_class_xlsx_format_srt;  by Name Gender;
run;
proc sort data=kids_class out=kids_class_srt;   by Name Gender; run;
data merged_class_xlsx_format;
merge boy_class_xlsx_format_srt(in=a)   kids_class_srt(in=b);
by Name Gender;
if a^=1 and b=1;
run;
/*good codes*/
data boy_class_xlsx_format;
format Name $ 11. Gender $ 6.;
set boy_class_xlsx;
run;
```

```
proc sort data=boy_class_xlsx_format(keep=Name) out=boy_class_xlsx_format_srt;  by
Name;   run;
proc sort data=kids_class out=kids_class_srt;   by Name; run;
data merged_class_xlsx_format;
merge boy_class_xlsx_format_srt(in=a)     kids_class_srt(in=b);
by Name;
if a=1 and b=1;
run;

/*proc import csv file*/
proc import out=boy_class_csv datafile="&path./kids_class.csv" dbms=csv replace;
delimiter=' ';   getnames=YES;
run;

/*good*/
data boy_class_csv_revar(rename=(Name_new=Name Gender_new=Gender));
set boy_class_csv;
length Name_new $ 11. Gender_new $ 6.;
Name_new = left(strip(Name));
Gender_new = strip(Gender);
```

```
drop Name Gender;

run;

proc sort data=boy_class_csv_revar out=boy_class_csv_revar_srt;   by Name;   run;

proc sort data=kids_class out=kids_class_srt;   by Name; run;

data merged_class_csv_revar;

merge boy_class_csv_revar_srt(in=a)   kids_class_srt(in=b);

by Name;   if a^=1 and b=1;   run;


/*bad*/

data boy_class_csv_relength;   length Name $ 11. Gender $ 6.;   set boy_class_csv;   run;

proc sort data=boy_class_csv_relength out=boy_class_csv_relength_srt;   by Name;   run;

proc sort data=kids_class out=kids_class_srt;   by Name; run;

data merged_class_csv_relength;

/*length Name $ 11. Gender $ 6.;   */

merge boy_class_csv_relength_srt(in=a)   kids_class_srt(in=b);

by Name;   if a^=1 and b=1;   run;


data merged_class_csv_relength_or;

/*length Name $ 11. Gender $ 6.;   */

merge boy_class_csv_relength_srt(in=a)   kids_class_srt(in=b);
```

```
by Name;    if a=1 or b=1;    run;


/*good*/
data boy_class_csv_format;    format Name $ 11. Gender $ 6.;    set boy_class_csv;    run;
proc sort data=boy_class_csv_format out=boy_class_csv_format_srt;    by Name;     run;
proc sort data=kids_class out=kids_class_srt;   by Name; run;
data merged_class_csv_format;
merge boy_class_csv_format_srt(in=a)   kids_class_srt(in=b);
by Name;    if a^=1 and b=1;    run;


/*proc export data=kids_class_srt outfile="&path./kids_class.txt"  dbms=csv replace;
delimiter=" ";
putnames=YES;  run;*/
/*proc export data=kids_class_srt outfile="&path./kids_class.csv" dbms=csv replace;
delimiter=" ";
putnames=YES;  run;*/


/*if the yellow part data was copied from excel file, Export from SAS EG
using Export on the top of SAS EG Program window It will cause mismatches.
Try to export as txt or csv files, then copy and paste here will work well*/
```

```
data boy_class_txt;
input Name $ 11.  Gender $ 7.  Sex $ 2. Age 8.;
datalines;
Alfred     MPle   M 14
Henry      MBle   M 14
Robert     Male   M 12
Robert     MRle   M 12
Thomas     MTle   M 11
Thomas     Male   M 11
William    Male   M 15
KaiqingFan MFle   M 12
run;


proc sort data=boy_class_txt out=boy_class_txt_revar_srt;   by Name;   run;
proc sort data=kids_class out=kids_class_srt;  by Name;   run;
data merged_class_txt_revar;
merge kids_class_srt(in=a)  boy_class_txt_revar_srt(in=b);
by Name;
if a=1 and b=1;
run;
```

```
proc sort data=boy_class_txt_revar out=boy_class_txt_revar1_srt;   by Name Gender;   run;

proc sort data=kids_class out=kids_class_srt1;  by Name Gender ;   run;

data merged_class_txt_revar;

merge kids_class_srt1(in=a)  boy_class_txt_revar1_srt(in=b);

by Name Gender;

/*if a^=1 and b=1;   */

run;


proc sort data=boy_class_txt_revar out=boy_class_txt_revar1_srt;   by Name ;   run;

proc sort data=kids_class out=kids_class_srt1;  by Name  ;   run;

data merged_class_txt_revar2;

merge kids_class_srt1(in=a)  boy_class_txt_revar1_srt(in=b);

by Name ;

/*if a^=1 and b=1;   */

run;


data boy_class_txt_relength;   set boy_class_txt;   run;

proc sort data=boy_class_txt_relength out=boy_class_txt_relength_srt;   by Name;   run;

proc sort data=kids_class out=kids_class_srt;                                   by Name;   run;
```

```
data merged_class_txt_relength;

length Name $ 11. Gender $ 6.;

merge boy_class_txt_relength_srt(in=a)   kids_class_srt(in=b);

by Name;   if a^=1 and b=1;   run;


data merged_class_txt_relength_or;

length Name $ 11. Gender $ 6.;

merge boy_class_txt_relength_srt(in=a)   kids_class_srt(in=b);

by Name;   if a=1 or b=1;   run;
```