PharmaSUG China 2016 – Paper 50

What's on behind – Revisit DATA step processing to avoid unexpected errors in a DATA step

Xuan Sun, PPD, Beijing, China

ABSTRACT

In our daily work, it is common to integrate information from multiple data sources. Benefited from DATA step, such manipulations could be handled efficiently. However, negligence of the DATA step processing flow may sometimes results in unexpected results. In this paper, the compilation and execution phases in DATA step processing will be illustrated. Also examples will be given to show the scenarios when we make mistakes.

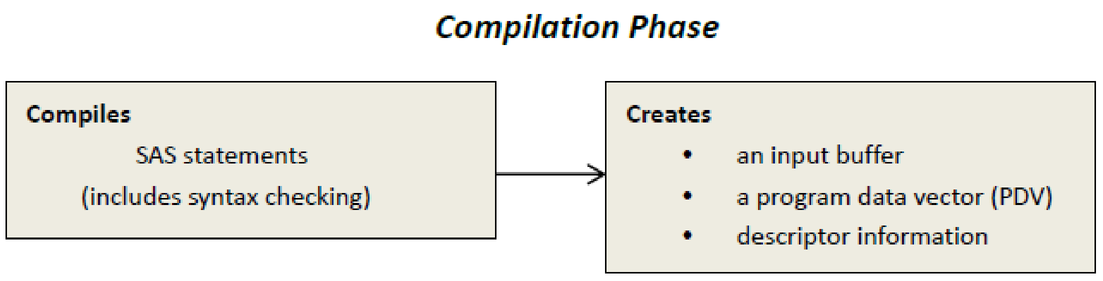
INTRODUCTION

When we submit a DATA step for execution, it is first compiled and then executed. The following data step processing flow and illustrations show the action for a typical SAS DATA step.

Compilation Phase

SAS checks the syntax of the SAS statements and compiles them, that is, automatically translates the statements into machine code. During the compilation phase, SAS creates the following three items:

1. **Input Buffer** is a logical area in memory into which SAS reads each record of raw data when SAS executes an INPUT statement. Note that this buffer is created only when the DATA step reads raw data not when reads a SAS dataset.
2. **Program Data Vector (PDV)** is a logical area in memory where SAS builds a dataset, one observation at a time. When a program executes, SAS reads data values from the input buffer or creates them by executing SAS language statements. Along with dataset variables and computed variables, the PDV contains two automatic variables, _N_ and _ERROR_. The _N_ variable counts the number of times the DATA step begins to execute. The _ERROR_ variable signals the occurrence of an error caused by the data during execution. The default value is 0, which means there is no error. When one or more errors occur, the value is set to 1.
3. **Descriptor Portion** is information that SAS creates and maintains about each SAS dataset. It contains, for example, the name of the dataset and its member type, the date and time that the dataset was created, and the number, names and data types (character or numeric) of the variables.

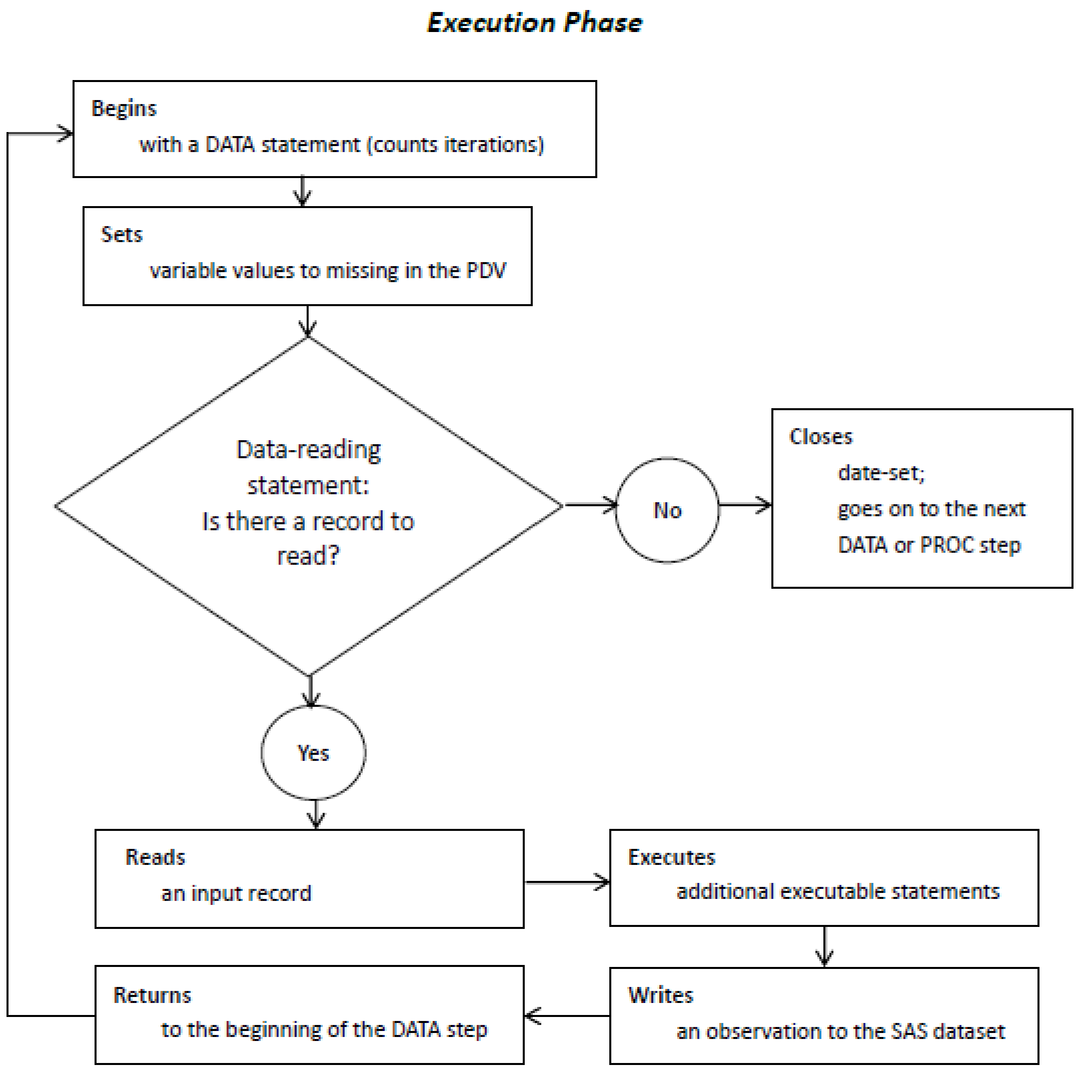


Execution phase

1. The DATA step begins with a DATA statement. Each time the DATA statement executes, a new iteration of the DATA step begins, and the _N_ automatic variable is incremented by 1.
2. SAS sets the newly created program variables to missing in the program data vector (PDV).
3. SAS reads a data record from a raw data file into the input buffer, or it reads an observation from a

SAS data set directly into the PDV.

4. SAS executes any subsequent programming statements for the current record.
5. At the end of the statements, an output, return, and reset occur automatically. SAS writes an observation to the SAS data set, the system automatically returns to the top of the DATA step, and the values of variables created by INPUT and assignment statements are reset to missing in the PDV. Note that variables that read with a SET, MERGE, MODIFY, or UPDATE statement are not reset to missing here.
6. SAS counts another iteration, reads the next record or observation, and executes the subsequent programming statements for the current observation.
7. The DATA step terminates when SAS encounters the end-of-file in a SAS data set or a raw data file.

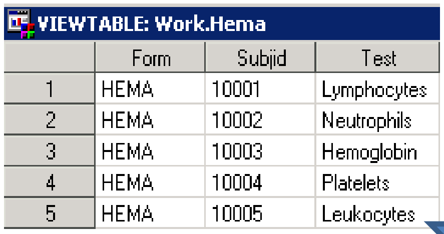

Execution Phase
Begins
with a DATA statement (counts iterations)
Sets
variable values to missing in the PDV
Data-reading statement: Is there a record to read?
No
Closes
date-set;
goes on to the next
DATA or PROC step
Yes
Reads
an input record
Executes
additional executable statements
Writes
an observation to the SAS dataset
Returns
to the beginning of the DATA step

We often merge, concatenate or interleave two or more datasets in one DATA step. And then do some further derivation after the combination. Negligence of the DATA step processing flow may sometimes lead to unexpected results.

The following two examples will show the scenarios when we make mistakes and what happens during the DATA step processing.

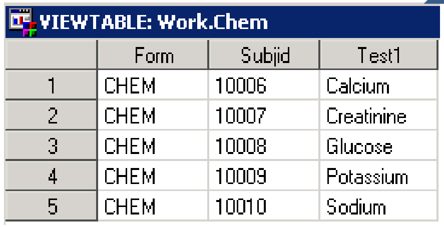
EXAMPLE 1

In this example, records from CHEM and HEMA are appended together. By the way, value of Variable TEST1 need to be assigned to TEST. Results expected are shown in dataset ALL.



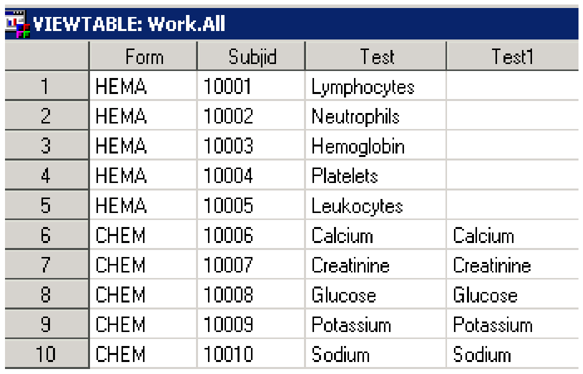
VIEWTABLE: Work.Hema

	Form	Subjid	Test
1	HEMA	10001	Lymphocytes
2	HEMA	10002	Neutrophils
3	HEMA	10003	Hemoglobin
4	HEMA	10004	Platelets
5	HEMA	10005	Leukocytes



VIEWTABLE: Work.Chem

	Form	Subjid	Test1
1	CHEM	10006	Calcium
2	CHEM	10007	Creatinine
3	CHEM	10008	Glucose
4	CHEM	10009	Potassium
5	CHEM	10010	Sodium

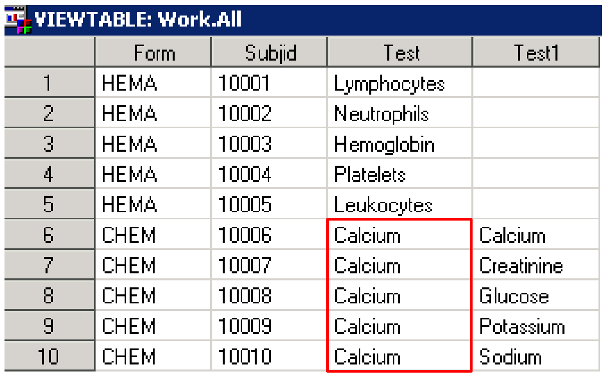


VIEWTABLE: Work.All

	Form	Subjid	Test	Test1
1	HEMA	10001	Lymphocytes	
2	HEMA	10002	Neutrophils	
3	HEMA	10003	Hemoglobin	
4	HEMA	10004	Platelets	
5	HEMA	10005	Leukocytes	
6	CHEM	10006	Calcium	Calcium
7	CHEM	10007	Creatinine	Creatinine
8	CHEM	10008	Glucose	Glucose
9	CHEM	10009	Potassium	Potassium
10	CHEM	10010	Sodium	Sodium

However, if we use the following codes for the purpose will produce undesirable results.

```
data all;
    set hema chem;
    if test='' and test1^='' then test=test1;
run;
```



VIEWTABLE: Work.All

	Form	Subjid	Test	Test1
1	HEMA	10001	Lymphocytes	
2	HEMA	10002	Neutrophils	
3	HEMA	10003	Hemoglobin	
4	HEMA	10004	Platelets	
5	HEMA	10005	Leukocytes	
6	CHEM	10006	Calcium	Calcium
7	CHEM	10007	Calcium	Creatinine
8	CHEM	10008	Calcium	Glucose
9	CHEM	10009	Calcium	Potassium
10	CHEM	10010	Calcium	Sodium

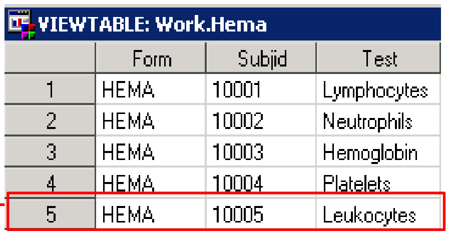
Reason for unexpected errors:

1. During the compilation phase, SAS reads the descriptor information of each dataset that is named in the SET statement and then creates a Program Data Vector (PDV) that contains all the variables from all datasets as well as variables created by the DATA step. The following variables in PDV are all from the two datasets and no created ones.

Program Data Vector

_N_	_ERROR_	Form	Subjid	Test	Test1

2. During the execution phase, SAS reads the first 5 observations from the first dataset HEMA into the PDV. It processes the observations and executes the IF-THEN clause in the DATA step. However, the IF-THEN statement is NULL for dataset HEMA. It then writes the contents of the PDV to the new dataset ALL. At the end of the fifth iteration, the values in the PDV are shown as below.



VIEWTABLE: Work.Hema

	Form	Subjid	Test
1	HEMA	10001	Lymphocytes
2	HEMA	10002	Neutrophils
3	HEMA	10003	Hemoglobin
4	HEMA	10004	Platelets
5	HEMA	10005	Leukocytes

_N_	_ERROR_	Form	Subjid	Test	Test1
5	0	HEMA	10005	Leukocytes	

3. The SET statement does not reset the values in the PDV to missing, except for variables whose value is calculated or assigned during the DATA step. But the values of the variables in the PDV are set to missing each time SAS starts to read a new dataset. Therefore, before the iterations in the second dataset CHEM, values of all variables in the PDV are set to missing. The value of _N_ is set to 6. Automatic variable _ERROR_ retains its value.

_N_	_ERROR_	Form	Subjid	Test	Test1
6	0				

4. The SET statement reads the sixth observation from dataset CHEM and writes the values to the PDV.

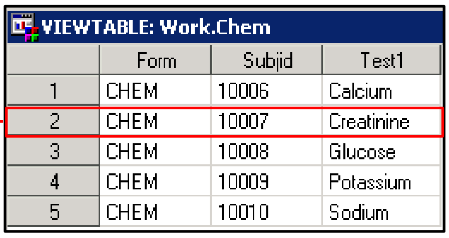
_N_	_ERROR_	Form	Subjid	Test	Test1
6	0	CHEM	10006		Calcium

Then, the IF-THEN statement executes to reassign the value for Variable TEST.

```
data all;
    set hema chem;
    if test='' and test1^='' then test=test1;
run;
```

_N_	_ERROR_	Form	Subjid	Test	Test1
6	0	CHEM	10006	Calcium	Calcium

5. SAS retains the values of variables that were read from the last observation and start the next iteration. However, dataset CHEM doesn't have variable TEST and only the variables FORM, SUBJID and TEST1 are replaced by new values. Such process is repeated until the end of the DATA step.



VIEWTABLE: Work.Chem

	Form	Subjid	Test1
1	CHEM	10006	Calcium
2	CHEM	10007	Creatinine
3	CHEM	10008	Glucose
4	CHEM	10009	Potassium
5	CHEM	10010	Sodium

_N_	_ERROR_	Form	Subjid	Test	Test1
7	0	CHEM	10007	Calcium	Creatinine

Solution:

The easiest way to correct the mistake is to RENAME the variable TEST1 to TEST in CHEM and no derivation afterwards needed.

```
**********The easitest solution***********;
data all;
    set hema chem(rename=(test1=test));
run;
```

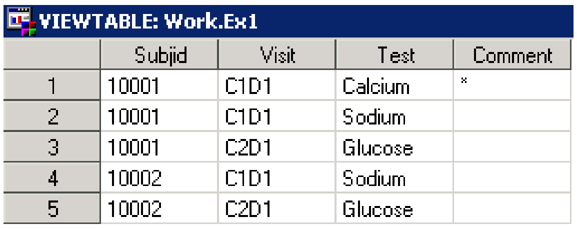
Here it is just an example to illustrate the potential causing errors. There are other two ways to correct the mistake.

1. Separate the DATA step into two steps, one for datasets combination and the other for the subsequent derivation. After combing to one dataset TEMP, every observation has the value of TEST even if some are missing ones. When the second DATA step executes, each variable in each observation during each iteration could be replaced by a new value.
2. We still can finish the work in one DATA step but we need to RENAME the variable TEST in HEMA to another one. Then both the two datasets don't have variable TEST and it becomes a new one which is assigned during the DATA step. The SET statement doesn't reset the values in the PDV to missing, except for variables whose values are calculated or assigned. So TEST is reset to missing in the PDV each time the iteration ends and it will not retain the value in the next iteration.

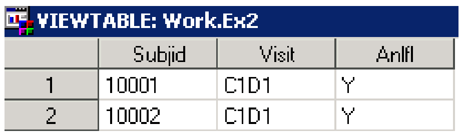
```
**********Solution1***********;
data temp;
    set hema chem;
run;

data all;
    set temp;
    if test='' and test1^='' then test=test1;
run;

**********Solution2***********;
data all(drop=test_o);
    set hema(rename=(test=test_o)) chem;
    test=test_o;
    if test_o='' and test1^='' then test=test1;
run;
```

EXAMPLE 2

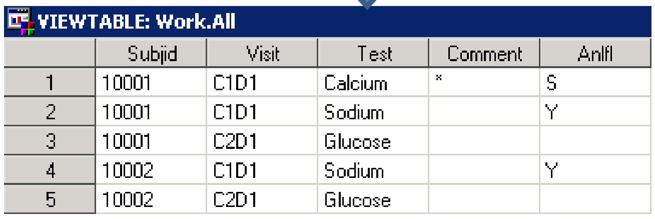
In this example, we need to merge ANLFL from EX2 on EX1 with key variables of SUBJID and VISIT. What's more, if Comment is '*', Value of ANLFL needs to be assigned to 'S'. Results expected is shown in dataset ALL.


VIEWTABLE: Work.Ex1

	Subjid	Visit	Test	Comment
1	10001	C1D1	Calcium	*
2	10001	C1D1	Sodium	
3	10001	C2D1	Glucose	
4	10002	C1D1	Sodium	
5	10002	C2D1	Glucose	


VIEWTABLE: Work.Ex2

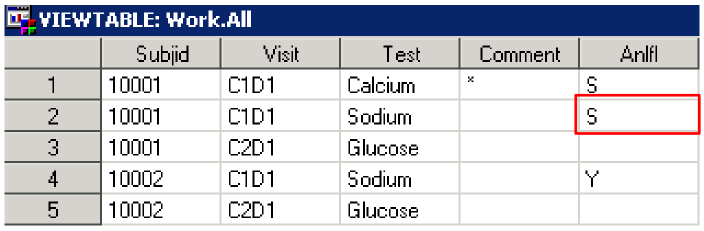
	Subjid	Visit	Anlfl
1	10001	C1D1	Y
2	10002	C1D1	Y


VIEWTABLE: Work.All

	Subjid	Visit	Test	Comment	Anlfl
1	10001	C1D1	Calcium	*	S
2	10001	C1D1	Sodium		Y
3	10001	C2D1	Glucose		
4	10002	C1D1	Sodium		Y
5	10002	C2D1	Glucose		

However, if we use the following codes for the purpose will produce undesirable results.

```
data all;
    merge ex1(in=a) ex2;
    by subjid visit;
    if a;
    if comment='*' then anlfl='S';
run;
```

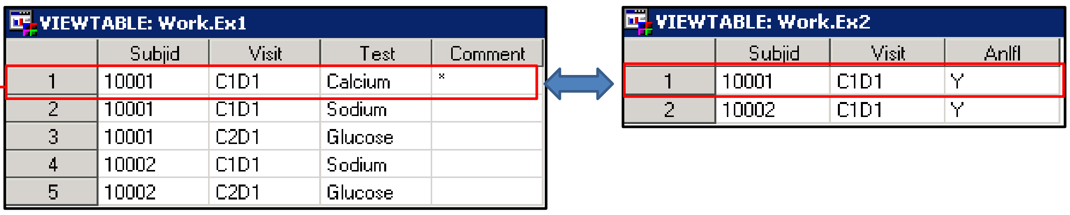


VIEWTABLE: Work.All

	Subjid	Visit	Test	Comment	Anlfl
1	10001	C1D1	Calcium	*	S
2	10001	C1D1	Sodium		S
3	10001	C2D1	Glucose		
4	10002	C1D1	Sodium		Y
5	10002	C2D1	Glucose		

Reason for unexpected errors:

1. During the compilation phase, SAS creates a PDV that contains all the variables. After compiling the DATA step, SAS sequentially match-merges observations by moving the pointers down each observation of each dataset and checking to see whether the BY values match. The first combined observation from EX1 and EX2 is written to the PDV as following.



VIEWTABLE: Work.Ex1

	Subjid	Visit	Test	Comment
1	10001	C1D1	Calcium	*
2	10001	C1D1	Sodium	
3	10001	C2D1	Glucose	
4	10002	C1D1	Sodium	
5	10002	C2D1	Glucose	

VIEWTABLE: Work.Ex2

	Subjid	Visit	Anlfl
1	10001	C1D1	Y
2	10002	C1D1	Y

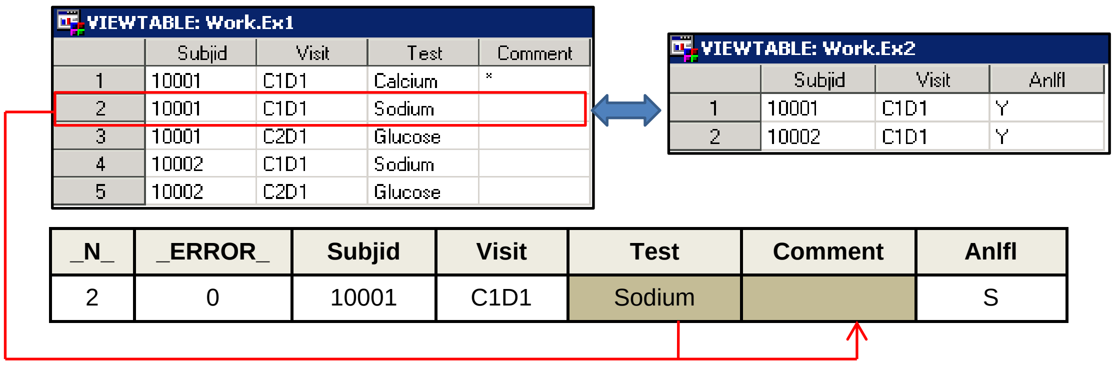
_N_	_ERROR_	Subjid	Visit	Test	Comment	Anlfl
1	0	10001	C1D1	Calcium	*	Y

2. SAS processes the observation and executes the IF-THEN clause in the DATA step. As the IF-THEN clause is true, the value of variable ANLFL is reassigned to 'S'. SAS writes the combined observation to the new dataset ALL and retains the values in the PDV until the BY values change in all the datasets.

```
data all;
    merge ex1(in=a) ex2;
    by subjid visit;
    if a;
    if comment='*' then anlfl='S';
run;
```

_N_	_ERROR_	Subjid	Visit	Test	Comment	Anlfl
1	0	10001	C1D1	Calcium	*	S

3. Before the second iteration beginning, the PDV retains its values because the BY values don't change. As the second observation from EX2 has BY values changed, SAS only writes the second observation from EX1 to the PDV. Therefore, only the variables TEST and COMMENT in the PDV are replaced by new values. So the combined observation has the value of ANLFL as 'S' again.



VIEWTABLE: Work.Ex1

	Subjid	Visit	Test	Comment
1	10001	C1D1	Calcium	*
2	10001	C1D1	Sodium	
3	10001	C2D1	Glucose	
4	10002	C1D1	Sodium	
5	10002	C2D1	Glucose	

VIEWTABLE: Work.Ex2

	Subjid	Visit	Anlfl
1	10001	C1D1	Y
2	10002	C1D1	Y

_N_	_ERROR_	Subjid	Visit	Test	Comment	Anlfl
2	0	10001	C1D1	Sodium		S

4. The BY values change in both EX1 and EX2 for the third iteration, so the PDV is initialized to missing.

_N_	_ERROR_	Subjid	Visit	Test	Comment	Anlfl
3	0					

Then SAS write the third observation to the PDV.

_N_	_ERROR_	Subjid	Visit	Test	Comment	Anlfl
3	0	10001	C2D1	Glucose		

5. The DATA step continues to process every observation in each dataset until it has processed all observations in all datasets.

Solution:

There are two ways to correct the mistake.

1. Separate the DATA step into two steps, one for datasets merging and the other for the subsequent derivation. When the second DATA step executes, each variable in each observation could be replaced by a new value.
2. Enhance the IF-THEN clause according to the PDV content in MERGE statement. But this method isn't recommended as the scenario may be very complex in practical work.

```
*********Solution1***********;
data temp;
    merge ex1(in=a) ex2;
    by subjid visit;
    if a;
run;

data all;
    set temp;
    if comment='*' then anlfl='S';
run;

**********Solution2***********;
data all;
    merge ex1(in=a) ex2;
    by subjid visit;
    if a;
    if comment='*' then anlfl='S';
    else if comment='' and anlfl='S' then anlfl='Y';
run;
```

CONCLUSION

In this paper, we discussed the DATA step processing flow and why sometimes we get undesirable results during a DATA step. Such mistakes sometimes are hidden from the code and hard to debug for new SAS users. The best way to avoid such mistakes is to separate the DATA step into two steps, one for datasets combination and the other for subsequent derivation.

To summarize, understanding the compilation phase and execution phase of the DATA step processing can help us better know what's on behind a DATA step.

REFERENCE

"SAS Certification Prep Guide Base Programing for SAS 9"

SAS 9.2 Online product documentation. Available at support.sas.com/documentation/92/index.html

ACKNOWLEDGEMENTS

The author would like to thank Lu Zhang for their insightful comments in reviewing an earlier publication of this paper.

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Name: Xuan Sun
Enterprise: PPD Inc.
Address: 25F, Raffles Business Center, No.1 Dongzhimen South Street, Dongcheng District
City, State ZIP: Beijing, 100007
Work phone: +86 10 61846112
E-mail: Xuan.Sun@ppdi.com
Web: www.ppdi.com