SESUG Paper 073-2021

Where Where is a Problem

Deb Cassidy, PPD

ABSTRACT

A statistician was reviewing a table and had different results than the programmers. Investigation showed the issue was really with the statistician's use of multiple WHERE's. This paper will show several ways of having multiple WHERE and IF statements. It will show which ones work as expected and which don't.

This is an entry level presentation. However, even long-time SAS programmers may learn something they never thought about. The author did. The code in the paper was tested using an interactive session of Enterprise Guide.

INTRODUCTION

A statistician was reviewing multiple related tables. The first table used the following code:

```
data check;
   set original_data(where=(result>2));
run;
```

The second table needed an additional restriction, so the statistician simply changed the code to:

```
data check;
   set original_data(where=(result>2));
       where pop='Y';
run;
```

This did not create the correct results. What happened and what should she have done?

TEST DATA

This is simple test data to show the issues.

POP	RESULT
Y	1
Y	2
Y	3
N	4
N	5

INITIAL RESULTS

The first data step returns 3 observations.

```
POP     RESULT

 Y         3
 N         4
 N         5
```

When the statistician added the second where to check the next table, she still had 3 observations instead of 1 observation which is correct. She then reported to the programmers that the count was incorrect on the table even though two programmers had independently obtained the result on the table.

```
POP     RESULT

 Y         3
 N         4
 N         5
```

However, she failed to do two things. She was only looking at the overall count so she didn't notice that the results included POP values that it shouldn't have. She could have done a couple quick checks to see if her results were reasonable.

She could have looked at her results with:

```
   proc freq data=check;
      tables POP*RESULT/list missing nocum nopercent;
   run;
```

```
POP     RESULT     Frequency
-------------------------
N            4             1
N            5             1
Y            3             1
```

She could have also looked at the original data to see what her count should have been:

```
   proc freq data=original_data;
      tables pop*result/list missing nocum nopercent;
   run;
```

```
POP     RESULT     Frequency
-------------------------
N            4             1
N            5             1
Y            1             1
Y            2             1
Y            3             1
```

Her other mistake which I consider a major mistake was not checking the log. The log made it obvious there was an issue and shows:

WARNING: The WHERE statement cannot be applied to the data set on the last SET/MERGE/UPDATE/MODIFY statement. Either the data set failed to open or it already specifies a WHERE data set option.

ALTERNATIVE 1

What if she changed and used 2 WHERE statements instead of a data set option and a statement?

```
data where_statements;
   set original_data;
   where RESULT>2;
   where POP='Y';
run;
```

This code returns these observations but that isn't right either. Why?

```
POP    RESULT

 Y        1
 Y        2
 Y        3
```

Once again, a quick check of the log yields the answer. When you have two where statements, the second one replaces the first one. The log shows:

NOTE: WHERE clause has been replaced.

ALTERNATIVE 2

What if she kept the WHERE dataset option but added an IF statement instead of a WHERE statement?

```
data where_if;
   set original_data (where=(RESULT>2));
   if POP='Y';
run;
```

This alternative does provide the correct results without any messages in the log.

```
POP    RESULT

 Y        3
```

ALTERNATIVE 3

WHERE ALSO is another option. This code returns the same incorrect result and log message as the original code so it doesn't fix the problem:

```
data where_also;
   set original_data(where=(RESULT>2));
   where also POP='Y';
run;
```

ALTERNATIVE 4

What happens if she changed Alternative 1 to use WHERE ALSO?

```
data where_statements_also;
   set original_data;
   where RESULT>2;
   where also POP='Y';
run;
```

You get the right results without any log messages.

```
POP     RESULT

 Y         3
```

ALTERNATIVE 5

What if she had chosen to use IF statements instead of WHERE? There isn't any problem with using 2 Ifs. They are both applied, obtain the expected results, and do not cause a log message.

```
data two_ifs;
   set original_data;
   if RESULT>2;
   if POP='Y';
run;
```

```
POP     RESULT

 Y         3
```

ALTERNATIVE 6

The statistician could have just used an "and" and included the second restriction as part of the original WHERE option. This also returns the correct result without any log issues.

```
   data where_and;
      set original_data(where=(RESULT>2 and POP='Y'));
   run;
```

```
POP     RESULT

 Y         3
```

CONCLUSION

One of the best things about SAS is the flexibility in how you write code. However, not everything works the way someone thinks it will. Always check your log for messages. It is also recommended that you do a quick check using another method.

So which alternatives worked as intended?

ALTERNATIVE	LOG MESSAGES	CORRECT RESULTS
WHERE dataset option & WHERE statement	Yes	No
2 WHERE statements	Yes	No
WHERE dataset option & IF statement	No	Yes
WHERE dataset option & WHERE ALSO statement	Yes	No
WHERE statement & WHERE ALSO statement	No	Yes
2 IF statements	No	Yes
WHERE dataset option using AND	No	Yes

CONTACT INFORMATION

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

Deb Cassidy
deborah.cassidy@ppd.com