Importance of Warnings and Notes messages from SAS log
Jyotheeswara Naidu Yellanki, Independent Consultant, Newark, DE

ABSTRACT:
When you run any SAS® code from any operating system, the first and most important rule in debugging SAS programs is to always check the SAS log. Generally you will look for only ERROR messages in the log. If there are no ERROR messages in the log you think that it is good to go and expect that results are good. It is advisable to check the warning and note message also. But you need to check Warnings and some kind of NOTE messages. Warnings are less dire than errors. SAS prints relevant warnings message in the log and then goes ahead and runs the job anyway. Many people, including some professional programmers, try to ignore warnings. Some times these warnings are Harmless or hazardous. Similarly SAS writes lot of NOTE messages into the log. Some NOTE messages tell about reasons for inaccurate results. This paper will describe some of the frequently occurring warnings and notes that are indicative of inaccurate results. In this article we will discuss a few warning and note messages.

INTRODUCTION
Validating a SAS program can be a tedious task. Generally SAS log files can have hundreds and even thousands of lines. Identifying the warning and note messages that are causing the inaccurate results is bit tough task. The tips provided in this paper will help to find out most frequently occurring warnings and note messages from log. This paper will give some examples from DATA step, few PROCs and Macros.

UNRESOLVED MACROS :
If a user defined macro does not exist or was misspelled when coded as shown below, then you will get a warning message:

```sas
%LET YR=2007;
Data new;
  Year="&Y";
Run;
WARNING: Apparent symbolic reference Y not resolved.
```

You can check the existence of a macro variable before using any macros variables by running the SAS function %SYMGLOBAL.

WHEN READING VARIABLE LENGTH RECORDS:
Following is an example of an external file with variable length records:

```
12339 A
24890 A 100
21137 B 43
14593 C
31003 C 67
```

The following DATA step reads the above external file. Because some records contain missing values, SAS will read from the next record in order to find a value for the variable and a message is written to the SAS log. This can produce unexpected results as shown in the following example:

```sas
data flowover;
infile 'external-file';
input id $ type $ amount;
proc print;
run;

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
```

Results of PROC PRINT for WORK.FLOWOVER:
<table>
<thead>
<tr>
<th>OBS</th>
<th>ID</th>
<th>TYPE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12339</td>
<td>A</td>
<td>24890</td>
</tr>
<tr>
<td>2</td>
<td>21137</td>
<td>B</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>14593</td>
<td>C</td>
<td>31003</td>
</tr>
</tbody>
</table>

**LOST CARD:**
The DATA step below reads the same external file from above but uses an informat of 3. to read in the values for AMOUNT. If the value for AMOUNT is missing or is not the exact length of 3 as specified by the informat, this can cause unexpected results as shown in the following example:

```
data flow2;
  infile 'external-file';
  input id $ type $ amount 3.;
  proc print;
  run;
```

Results of PROC PRINT for WORK.FLOW2:

<table>
<thead>
<tr>
<th>OBS</th>
<th>ID</th>
<th>TYPE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12339</td>
<td>A</td>
<td>248</td>
</tr>
<tr>
<td>2</td>
<td>21137</td>
<td>B</td>
<td>145</td>
</tr>
</tbody>
</table>

**REMERGING THE DATA:**
In PROC SQL if you want summarize the data by grouping the columns and calculate the statistics, the code is as shown below:

```
PROC SQL;
  SELECT SEX, COUNT(*) AS CNT
  FROM SASHELP.CLASS
  GROUP BY SEX;
  QUIT;
```

The results looks like

<table>
<thead>
<tr>
<th>Sex</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>9</td>
</tr>
<tr>
<td>M</td>
<td>10</td>
</tr>
</tbody>
</table>

When you use a summary function in a SELECT clause or HAVING clause without GROUP BY you will get the following message in the SAS log.

NOTE: The query requires remerging summary statistics back with the original data.

<table>
<thead>
<tr>
<th>Sex</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>19</td>
</tr>
<tr>
<td>F</td>
<td>19</td>
</tr>
<tr>
<td>M</td>
<td>19</td>
</tr>
<tr>
<td>F</td>
<td>19</td>
</tr>
<tr>
<td>M</td>
<td>19</td>
</tr>
<tr>
<td>F</td>
<td>19</td>
</tr>
<tr>
<td>M</td>
<td>19</td>
</tr>
<tr>
<td>M</td>
<td>19</td>
</tr>
</tbody>
</table>
SQL SYNTAX CHECK:
In the following PROC SQL, the option NOEXEC is used to validate the SQL syntax without executing the code. Some time you may forget to take out this option to release from syntax check mode. When you want really execute the code, it will display the note message in the log as shown below.

```
proc sql noexec;
select age,count(*)
from sashelp.class
group by age;
quit;
```

NOTE: Statement not executed due to NOEXEC option.

UNINITIALIZED VARIABLES:
If the variable var1 is not in dataset SASHELP.CLASS and not defined earlier then it will display the NOTE message in log. The IF condition will always be true and produces wrong results.

```
data new1;
set sashelp.class;
if age > var1 then group='teen age';
run;
```

NOTE: Variable var1 is uninitialized.

INVALID ARGUMENTS IN THE FUNCTIONS:
In the below code, if we supply invalid arguments to the INPUT function then it will display a note message in the SAS Log:

```
data new2;
x=input('20071142',yyymmd8.);
run;
```

NOTE: Invalid argument to function INPUT at line XX column YY.

NON-REFERENCED VARIABLES:
If you try to KEEP or DROP a variable, which is not defined, then SAS will display a warning message in the log.

```
data new2(keep=name age x);
set sashelp.class;
run;
```

WARNING: The variable x in the DROP, KEEP, or RENAME list has never been referenced.

LIMIT THE NUMBER OF OBSERVATIONS IN DATA STATEMENT:
If you are trying to use the OBS option on the Data statement, then you will get a warning message in the log.

```
data new2(obs=5);
set sashelp.class;
run;
```

WARNING 70-63: The option OBS is not valid in this context. Option ignored.

DIFFERENT LENGTHS FOR BY VARIABLE IN MERGE:
If emp_id in dataset one is char type and length 1 and emp_id in the dataset two is char type and length two and the data looks like below:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>emp_id</td>
<td>dep_cd</td>
<td>emp_id</td>
<td>dep_cd</td>
</tr>
<tr>
<td>4</td>
<td>AA</td>
<td>4</td>
<td>AM</td>
</tr>
<tr>
<td>5</td>
<td>AB</td>
<td>7</td>
<td>DE</td>
</tr>
<tr>
<td>6</td>
<td>AC</td>
<td>5</td>
<td>AF</td>
</tr>
<tr>
<td>7</td>
<td>AD</td>
<td>8</td>
<td>ED</td>
</tr>
</tbody>
</table>

Merge these two datasets and we will get a warning message in the log:

```sas
data three;
merge one two;
by emp_id;
run;
```

WARNING: Multiple lengths were specified for the BY variable emp_id by input data sets. This may cause unexpected results.

The output looks like:

<table>
<thead>
<tr>
<th>emp_id</th>
<th>dep_cd</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AA</td>
</tr>
<tr>
<td>5</td>
<td>BB</td>
</tr>
<tr>
<td>6</td>
<td>AC</td>
</tr>
<tr>
<td>7</td>
<td>AD</td>
</tr>
<tr>
<td>8</td>
<td>CD</td>
</tr>
</tbody>
</table>

DUPLICATE BY VALUES:
In the following example, dataset one and two have BY variable ID and both have duplicate values. When you merge these two datasets and execute the code, a note message is written to the SAS log.

```sas
data one;
ID=2; A=1; output;
ID=5; A=3; output;
ID=5; A=6; output;
ID=5; A=7; output;
run;

data two;
ID=2; C=6; output;
ID=5; C=2; output;
ID=5; C=8; output;
run;

** assumes data is already sorted;

data three;
merge one two;
by id;
run;
```

NOTE: MERGE statement has more than one data set with repeats of BY values.

These results are:

<table>
<thead>
<tr>
<th>Obs</th>
<th>ID</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
DIVISION BY ZERO:
In the following example, an execution-time error occurs when SAS uses data values from the second observation to
perform the division operation in the assignment statement. Division by 0 is an illegal mathematical operation and
causes an execution-time error.

```sas
data inventory;
  input Item $ 1-14 TotalCost 15-20
       UnitsOnHand 21-23;
  UnitCost=TotalCost/UnitsOnHand;
datalines;
Hammers       440   55
Nylon cord    35    0
Ceiling fans  1155  30
;
NOTE: Division by zero detected at line xx column yy.
```

MISSING VALUES:
A very common task in a DATA step is to sum the values of various variables. In this example the value of variable A
is missing and resultant sum will also be missing. When the code executes a note message is written to SAS log.

```sas
data abc;
  A=.; B=4; C=7;
  Total = A+B+C;
run;
NOTE: Missing values were generated as a result of performing an operation on
missing values.
```

DATA STEP LOOPING:
If a DATA step is written such that no data reading statements (e.g. SET, INPUT) are executed, the step is
terminated after one iteration and the Note message is written to the SAS log. For example, in the following DATA
step the INPUT statement is conditionally executed. At the time the INPUT statement executes, X's value is
missing. Since no data is read in, the step terminates.

```sas
data test;
  if x=1 then input y;
  x=1;
cards;
123
456
;
NOTE: DATA STEP stopped due to looping.
```

CONCLUSION
This paper addressed different Warning and Note messages from SAS Log that need closer attention. In addition we
discussed the scenarios why those messages occurred and when they will occur. The tips and examples discussed
in this paper will help you to avoid common problems, or to know how to recognize and rectify them when they occur.

REFERENCES

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

Jyotheswara Naidu Yellanki
Email: yjnaidu@hotmail.com

SAS and all other SAS Institute Inc., product or service names are registered trademarks or trademarks of SAS
Institute Inc. in the USA and other countries. ® Indicates USA registration.
Other brand and product names are trademarks of their respective companies.