Variable List Short-Cuts in PROC SQL

ABSTRACT

PROC SQL queries require the SELECT statement to specify the variables to be included in the output destination. You have the option to explicitly state which variables to keep or use the * wildcard as a short-cut to select all fields from some or all tables involved in the query. Have you ever encountered the problem of having to choose between typing in a long list of variables to keep or use the * to select more variables than are actually needed? Have you ever wanted to select all the fields from a table except for one or two fields? Using the DROP and KEEP data set options are particularly useful for alleviating these conundrums. Moreover, these data set options allow you to use variable list short-cuts that are not available in the SELECT statement.

INTRODUCTION

PROC SQL queries require the SELECT statement to specify the variables to be included in the output destination. Figure 1 below demonstrates the use of the * wildcard as a short-cut notation to select all the fields from the SCORES data set. Not displayed in the fictitious clinical trial data set SCORES are the character fields B5 through B10. The program that created the data set can be found in the Appendix at the end of this paper.

Figure 1 – SELECT * Short-Cut

<table>
<thead>
<tr>
<th>Subject_ID</th>
<th>Visit</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.18</td>
<td>0.80</td>
<td>2.76</td>
<td>2.17</td>
<td>0.25</td>
<td>4.92</td>
<td>5.97</td>
<td>7.66</td>
<td>2.45</td>
<td>9.77</td>
<td>1.47</td>
<td>1.53</td>
<td>9.23</td>
<td>9.03</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.69</td>
<td>1.12</td>
<td>1.43</td>
<td>2.54</td>
<td>2.91</td>
<td>4.37</td>
<td>6.52</td>
<td>4.72</td>
<td>3.52</td>
<td>6.80</td>
<td>0.91</td>
<td>1.64</td>
<td>8.10</td>
<td>9.46</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.17</td>
<td>0.00</td>
<td>2.70</td>
<td>0.20</td>
<td>2.56</td>
<td>1.06</td>
<td>2.63</td>
<td>3.63</td>
<td>5.17</td>
<td>4.40</td>
<td>1.37</td>
<td>4.23</td>
<td>5.61</td>
<td>2.65</td>
</tr>
</tbody>
</table>

proc sql;
create table Scores1 as
   select T1.*
   from Scores as T1
   where visit=1;
quit;

Partial display of the SCORES data set

The short-cut * in the SELECT statement means keep all uniquely named variables.

SELECT * EXCEPT, DON'T YOU WISH?

You have the option to explicitly state which variables to keep or use the * wildcard as a short-cut to select all fields from some or all tables involved in a PROC SQL query. Have you ever encountered the problem of having to choose between typing in a long list of variables to keep or use the * to select more variables than are actually needed? A construct like SELECT * EXCEPT would seem to be handy, but it does not exist. However, the dilemma can be alleviated by using the KEEP and DROP data set options in the FROM clause. These data set options allow you to specify which of the variables should be processed and those that should not, respectively. For example, suppose that all variables except A1 and A3 are needed from the SCORES data set. The query could be written succinctly by using the * wildcard and the DROP data set option to remove the fields as the table is introduced into the query, as shown on the next page in Figure 2.
Proc sql;
create table Scores1 as
select *
from Scores(drop=A1 A3);
quit;

More variable list short-cuts

One of the useful aspects of the implementation of SQL in the SAS® System is the availability of the short-cut notations that can be used with data set options. Suppose that all the fields from the SCORES data set were required except for the ten B parameters. The query could be written succinctly in a variety of ways by making use of any of the four short-cut notations shown below in Figure 3.

proc sql;
create table only_As_1(label='Scores for A Parameters Only') as
select *
from Scores(keep=Subject_id Visit A1-A10);
/* or */
create table only_As_2(label='Scores for A Parameters Only') as
select *
from Scores(drop=B1--B10);
/* or */
create table only_As_3(label='Scores for A Parameters Only') as
select *
from Scores(drop= B:);
/* or */
create table only_As_4(label='Scores for A Parameters Only') as
select *
from Scores(keep=Subject_ID _numeric_);
quit;

Note the use of the numbered range list (-) with the KEEP data set option reference in the first query. This short-cut refers to a list of variables with a common prefix with an indexed integer suffix. Specifying A1-A10 in the SELECT statement would have created a new variable that is the result of arithmetic subtraction of the two fields. The second query makes use of the name range list (--) with the DROP data set option to remove all variables in the data set located between B1
and B10. Note the use of the name prefix list courtesy of the colon in the third query to drop all variables beginning with the letter B. The fourth query demonstrates the use of the _numeric_variable class list keyword to reference all of the numeric fields in the SCORES data set. It is left to you to infer as to how the query could have been written using the _character_keyword. All of the aforementioned coding short-cuts are not valid in the SELECT statement but they are available when using the DROP and KEEP data set options in the FROM clause.

The DROP and KEEP data set options are also valid in the CREATE TABLE statement of PROC SQL. You may find that you need to delay the removal of some variables from the output destination because they are needed as part of join conditions or calculated fields. Consider the query below in Figure 4 where the sum of the B1 and B2 fields are needed for visits one through ten, along with the A fields.

```
proc sql;
create table Scores1(drop=B:) as
select *
  , sum(input(B1,8.3), input(B2,8.3)) as S_B1_B2
from Scores(drop=B3--B10)
where visit between 1 and 10;
quit;
```

The fields B3 through B10 are immediately pruned from the query due to the DROP data set option in the FROM clause. The remaining original fields from SCORES are selected with the * wildcard, as well as the newly calculated field S_B1_B2. Since it is not necessary to keep the fields B1 and B2 in the output table, they are omitted from the table SCORES1 via the DROP data set option in the CREATE TABLE statement.

LIMITATIONS

There are limitations to using the very handy DROP and KEEP data set options in PROC SQL, including the following:

- They are not valid in the FROM clause of queries against the SAS system (metadata) DICTIONARY tables.
- Since the metadata SASHELP views are created with PROC SQL from the DICTIONARY tables, data set options are not valid in the FROM clause in queries against them.
- They are not valid in the FROM clause against views created with PROC SQL. However, using the data set options are valid in the FROM clause for views created with a DATA step.
- They are not applied when used in the CREATE VIEW statement, and you will receive a warning indicating so.
- Data set options are not valid syntax when querying a non-SAS relational database such as ORACLE and DB2 with the PROC SQL Pass-Through facility. They are valid when querying non-SAS relational databases when the connection to the tables is established with a LIBNAME.
CONCLUSION

This paper has demonstrated how using the KEEP and DROP data set options in PROC SQL queries can be used to remove or select a subset of fields while still making use of the * wildcard short-cut. In addition, these data set options allow you to use variable list short-cuts that are not available in the SELECT statement, such as the name and numbered range lists. These tips should help you minimize the number of keystrokes needed to keep unnecessary fields out of your data sets.

REFERENCES


SAS V9 On-Line Document

ACKNOWLEDGEMENTS

The author would like to thank Jenni Borowiak and Melissa Yott for their insightful comments in reviewing this paper.

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.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:
Kenneth W. Borowiak
Howard M. Proskin & Associates, Inc.
300 Red Creek Drive, Suite 220
Rochester, NY 14623

E-mail: KBorowiak@hmproskin.com
EvilPettingZoo97@aol.com

APPENDIX 1 – Code to Generate Sample Data Set

data Scores(label='Recorded Scores at Visits' drop=j c);
length Subject_ID $ 8 Visit 8 A1-A10 8 B1-B10 $ 8;
array A[10]; array B[10];
do c=1 to 3000;
   if c le 1500 then subject_id=compress('100-'||put(c,z4.));
   else subject_id=compress('200-'||put(c, z4.));
do Visit=1 to 20;
   do j=1 to 10;
      a[j]=j*ranuni(j);
      b[j]=left(put(.5+j**2*ranuni(j),7.3));
   end;
   output;
end;
format a: 5.2;
run;

NOTE: The data set WORK.SCORES has 60000 observations and 22 variables.