Chapter 1 PERFORMING QUERIES USING PROC SQL

Reference

SAS Certification Prep Guide: Advanced Programming for SAS 9, SAS Publishing Inc., 3rd ed., 2011, ISBN: 978-1607649250

**PROC SQL** step provides an efficient way to retrieve information from data sets. The abbreviation SQL stands for **Structured Query Language**. There are several statements that can be used with the PROC SQL step. We will study the **SELECT statement**. A PROC SQL step that contains one or more SELECT statements is called a **PROC SQL query**.

THE SELECT STATEMENT

The syntax is as follows:

PROC SQL;

SELECT *variable1*, *variable2*, … <**as** *newvariable>*

FROM *dataset1, dataset2 …*

WHERE *expression*

GROUP BY *variable1, variable2* …

HAVING *expression*

ORDER BY *variablename1, variablename2 …***;**

QUIT;

* SELECT specifies column(s) to be selected
* FROM specifies table(s) to be queried
* WHERE subsets the data based on the specified expression
* GROUP BY groups data by specified variable(s)
* HAVING subsets the data based on the specified expression involving statistics on individual variables
* ORDER BY classifies the data into groups based on the specified column(s)
* The SELECT is a statement is composed of **clauses**: FROM, WHERE, GROUP BY, HAVING, ORDER BY. These clauses must appear in the specified order. The sentence that help remembering the order is “So Few Workers Go Home On-time”.
* Date sets listed under the FROM clause are referred to as **tables**.
* To terminate PROC SQL, the **QUIT** statement must be specified.

Example. Class roster contains students’ names, IDs, scores on the final, total scores and letter grades. The following code uses PROC SQL to select the students with passing grades and to sort the grade roster in alphabetical order.

data roster;

input name $ @11 id $ final total grade $;

cards;

Hano 9336 92 98 A

Kelbert 9564 96 95 A

Le 2143 98 93 A

Allen 5686 83 92 A

Chen 1414 67 90 A

Chua 7573 82 89 B

Disbrow 7497 58 88 B

Seo 5666 65 88 B

Hogan 7730 64 87 B

Lacap 6734 83 84 B

Jin 8307 71 84 B

Ciralli 7002 53 78 C

Karni 8793 65 78 C

Martinez 7057 28 77 C

Davis 5392 58 77 C

Franjic 8455 0 53 I

Shimazaki 7439 40 63 D

Uy 4186 0 20 F

;

proc sql;

select name, grade

from roster

where total>=60

order by name;

quit;

The output is

name grade

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Allen A

Chen A

Chua B

Ciralli C

Davis C

Disbrow B

Hano A

Hogan B

Jin B

Karni C

Kelbert A

Lacap B

Le A

Martinez C

Seo B

Shimazak D

QUERYING MULTIPLE TABLES

PROC SQL can be used to combine selected information from files. The procedure is called **querying multiple tables**.

Example. The homeworks data set contains student ids and scores on homeworks 1 and 2. The homework3 data set contains student ids and scores on homework 3. Some students with low scores on the first two homeworks have dropped the course. The code below merges the two data sets with respect to student ids, computes the average of the three homeworks for every student, and sorts the entries in increasing order with respect to student ids.

data homeworks;

input id $ hw1 hw2;

cards;

2157 93 85

8166 86 80

4499 100 90

2852 100 60

8820 65 60

6454 93 70

931 65 25

3158 70 0

2839 93 80

;

data homework3;

input id $ hw3;

cards;

2157 78

8166 91

4499 84

2852 88

6454 62

2839 73

;

proc sql;

select homeworks.id, hw1, hw2, hw3,

mean(hw1, hw2, hw3) as average

from homeworks, homework3

where homeworks.id=homework3.id

order by homeworks.id;

quit;

* Both data sets contain the variable id. To tell them apart, use the data set name as a prefix, that is, refer to id as homeworks.id and homework3.id, respectively.
* mean(hw1, hw2, hw3) as average computes the mean of hw1-hw3 and names the new variable average.

The output is

id hw1 hw2 hw3 average ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2157 93 85 78 85.33333

2839 93 80 73 82

2852 100 60 88 82.66667

4499 100 90 84 91.33333

6454 93 70 62 75

8166 86 80 91 85.66667

CREATING OUTPUT TABLES WITH PROC SQL

* To output a new data set with PROC SQL, use the syntax

PROC SQL;

CREATE TABLE *tablename* as

SELECT statement;

QUIT;

Example. In the previous example, a new data file called roster is created. It is stored in the WORK library.

proc sql;

create table roster as

select homeworks.id, hw1, hw2, hw3,

mean(hw1, hw2, hw3) as average

from homeworks, homework3

where homeworks.id=homework3.id

order by homeworks.id;

quit;

proc print;

run;

The output is

Obs id hw1 hw2 hw3 average

1 2157 93 85 78 85.3333

2 2839 93 80 73 82.0000

3 2852 100 60 88 82.6667

4 4499 100 90 84 91.3333

5 6454 93 70 62 75.0000

6 8166 86 80 91 85.6667

* PROC SQL can provide distinct values of specified variables.

Example. The following data contain patient id, patient’s gender, the number of the visit to the doctor’s office, and the dosage of a prescribed medication. PROC SQL is employed to list only the distinct patient ids.

data dosages;

input ptID $ gender $ visit dose;

cards;

101 M 1 20

101 M 2 30

101 M 3 40

101 M 4 55

102 F 1 35

102 F 2 45

103 F 1 15

103 F 2 20

103 F 3 25

104 M 1 20

104 M 2 35

105 M 1 25

105 M 2 30

105 M 3 30

106 F 1 35

106 F 2 45

;

proc sql;

select distinct ptID

from dosages;

quit;

The output is

ptID

ƒƒƒƒƒƒƒƒ

101

102

103

* PROC SQL can provide the count of the number of distinct values of specified variables.

Example. In the previous example, the following code produces the number of distinct values of patient id.

proc sql;

select count (distinct ptID) as mycount

from dosages;

quit;

The output is

count

ƒƒƒƒƒƒƒƒ

3

* PROC SQL can provide a summary of data grouped by a specified variable. The clause GROUP BY must be used in the SELECTION statement.

Example. In the previous example, the following code counts distinct values of patient id for each gender.

proc sql;

select gender, count (distinct ptID) as count

from dosages

group by gender;

quit;

The output is

gender count

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

F 2

M 1

* PROC SQL can compute the number of times each distinct value appears.

Example. In the previous example, the following code counts distinct values of visits for each patient id.

proc sql;

select ptID, count(visit) as nvisits

from dosages

group by ptID;

quit;

The output is

ptID nvisits

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

101 4

102 2

103 3

To produce frequency distribution for the number of visits, run the following code:

proc sql;

create table visitsdata as

select ptID, count(visit) as nvisits

from dosages

group by ptID;

select \* from visitdata;

quit;

| **ptID** | **nvisits** |
| --- | --- |
| 101 | 4 |
| 102 | 2 |
| 103 | 3 |
| 104 | 2 |
| 105 | 3 |
| 106 | 2 |

proc sql;

select nvisits, count(\*)

from visitsdata

group by nvisits;

quit;

| **nvisits** | **npatients** |
| --- | --- |
| 2 | 3 |
| 3 | 2 |
| 4 | 1 |