PROC SQL Tips and Tricks

Suzanne Humphreys
The Rick Hansen Institute

Overview

- PROC SQL adds many powerful and useful features to SAS
- Mix BASE SAS and SQL syntax
- My 10 favourite SQL tips (except there are only 9 because I can’t count)

The Select Statement Syntax

- The purpose of the SELECT statement is to describe how the report will look.
- It consists of the SELECT clause and several sub-clauses.
- The sub-clauses name the input dataset, select rows meeting certain conditions (subsetting), group (or aggregate) the data, and order (or sort) the data:

PROC SQL *options*;
SELECT *column(s)*
FROM *table-name* | *view-name*
WHERE *expression*
GROUP BY *column(s)*
HAVING *expression*
ORDER BY *column(s);*

Statements Supported by PROC SQL

- In addition to the SELECT statement, PROC SQL supports the following statements:

- **ALTER TABLE**: add, drop, and modify columns in a table
- **CREATE**: build new tables, views, or indexes
- **DELETE**: eliminate unwanted rows from a table or view
- **DESCRIBE**: display table and view attributes
- **DROP**: eliminate entire tables, views, or indexes
- **INSERT**: add rows of data to tables or views
- **RESET** <option(s)>: add to or change PROC SQL options without re-invoking the procedure
- **UPDATE**: modify data values in existing rows of a table or view

Summary Functions Supported by PROC SQL

- **AVG, MEAN**: mean or average of values
- **COUNT, FREQ, N**: number of non missing values
- **CSS**: corrected sum of squares
- **CV**: coefficient of variation (percent)
- **MAX**: largest value
- **MIN**: smallest value
- **NMISS**: number of missing values
- **PRT**: probability of a greater absolute value of Student's t
- **RANGE**: range of values
- **STD**: standard deviation
- **STDERR**: standard error of the mean
- **SUM**: sum of values
- **T**: student's t value
- **USS:** uncorrected sum of squares
- **VAR**: variance

Good Overview of Basic PROC SQL Syntax

Katie Minten Ronk, Steve First, David Beam

AN INTRODUCTION TO PROC SQL®

http://www2.sas.com/proceedings/sugi27/p191-27.pdf

Summarizing Data Down Rows

- Aggregate results straight forward and quicker in PROC SQL

```
PROC SQL;
  select Patient, Trt_Grp, Height
  from HeightData;
QUIT;
```

Patient	Trt_Grp	Height
1	1	155.2
2	3	140.1
3	1	162.1
4	2	154.0
5	2	180.6

etc.

Summarizing Data Down Rows

```
PROC SQL;
  select AVG(Height) as Average_Height
   from HeightData
   where Trt_Grp=1;
QUIT;
```

```
Average_Height
 161.0765
```

Creating Macro Variables with Aggregate Functions

```
PROC SQL;
 select count(*) as Cnt, group
 from sashelp.bmt
 group by group;
QUIT;
```

Cnt	Disease Group
38	ALL
45	AML-High Risk
54	AML-Low Risk

Creating Macro Variables with Aggregate Functions

```
PROC SQL;
  select Cnt
         into: tot1 - : tot3
  from (select count(*) as Cnt, group
        from sashelp.bmt
        group by group);
QUIT;

%put TOT1=&tot1 TOT2=&tot2 TOT3=&tot3;
TOT1=38 TOT2=45 TOT3=54
```

Creating Macro Vars for a %Do Loop

- SASHELP.CARS, first 16 observations

Make	Model	Type	Origin	MSRP
Acura	MDX	SUV	Asia	$36,945
Acura	RSX Type S 2dr	Sedan	Asia	$23,820
Acura	TSX 4dr	Sedan	Asia	$26,990
Acura	TL 4dr	Sedan	Asia	$33,195
Acura	3.5 RL 4dr	Sedan	Asia	$43,755
Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	$46,100
Acura	NSX coupe 2dr manual S	Sports	Asia	$89,765
Audi	A4 1.8T 4dr	Sedan	Europe	$25,940
Audi	A41.8T convertible 2dr	Sedan	Europe	$35,940
Audi	A4 3.0 4dr	Sedan	Europe	$31,840
Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	$33,430
Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	$34,480
Audi	A6 3.0 4dr	Sedan	Europe	$36,640
Audi	A6 3.0 Quattro 4dr	Sedan	Europe	$39,640
Audi	A4 3.0 convertible 2dr	Sedan	Europe	$42,490
Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	$44,240

Creating Macro Vars for a %Do Loop

```
PROC SQL noprint;
   select distinct make, count(distinct make)
   into :make separated by '*',
        :makecnt
   from sashelp.cars;
QUIT;
```

%put MAKE=&make MAKECNT =&makecnt;

```
MAKE=Acura*Audi*BMW*Buick*Cadillac*Chevrolet*Chrysler*Dodge
*Ford*GMC*Honda*Hummer*Hyundai*Infiniti
*Isuzu*Jaguar*Jeep*Kia*Land Rover*Lexus*Lincoln*MINI*
Mazda*Mercedes-Benz*Mercury*Mitsubishi*Nissan*Oldsmobile
*Pontiac*Porsche*Saab*Saturn*Scion*Subaru*Suzuki*Toyota
*Volkswagen*Volvo
MAKECNT = 38
```

Creating Macro Vars for a %Do Loop

```
%macro dummy;
%do i=1 %to &makecnt;
  data %scan(%scan(&make,&i,*),1,-);
    set sashelp.cars
          (where=(make="%scan(&make,&i,*)"));
%end;
%mend dummy;
%dummy
run;
```

- Results in 38 data sets named after each make of car, containing only data for that make.

Simulating Lags and Leads

- Example Input:

ID	YEAR	X1
1	1990	0.3
1	1991	0.45
1	1992	0.4
1	1993	0.25
1	1994	0.5
3	1999	1.1
3	2000	1.3
3	2001	2.45
3	2002	3.4
3	2003	2.5
3	2004	4.5

Simulating Lags and Leads

```
PROC SQL;
create table want as
 select distinct
  have.*,
       sum(case when have.year-1 = cross.year then cross.x1
                else . end) as lag1x1,
       sum(case when have.year-2 = cross.year then cross.x1
                else . end) as lag2x1,
       sum(case when have.year-3 = cross.year then cross.x1
                else . end) as lag3x1
 from have
 inner join
 have as cross
 on have.id = cross.id
 group by have.id, have.year;
QUIT;
```

Simulating Lags and Leads

- Output (lag & lead of X1):

ID	YEAR	X1	LAG1X1	LAG2X1	LAG3X1	LEAD1X1	LEAD2X1	LEAD3X1
1	1990	0.3	.	.	.	0.45	0.4	0.25
1	1991	0.45	0.3	.	.	0.4	0.25	0.5
1	1992	0.4	0.45	0.3	.	0.25	0.5	.
1	1993	0.25	0.4	0.45	0.3	0.5	.	.
1	1994	0.5	0.25	0.4	0.45	.	.	.
3	1999	1.1	.	.	.	1.3	2.45	3.4
3	2000	1.3	1.1	.	.	2.45	3.4	2.5
3	2001	2.45	1.3	1.1	.	3.4	2.5	4.5
3	2002	3.4	2.45	1.3	1.1	2.5	4.5	.
3	2003	2.5	3.4	2.45	1.3	4.5	.	.
3	2004	4.5	2.5	3.4	2.45	.	.	.

Simulating Lags and Leads

```
PROC SQL;
create table want as
 select distinct
  have.*,
       sum(case when have.year+1 = cross.year then cross.x1
                else . end) as lead1x1,
       sum(case when have.year+2 = cross.year then cross.x1
                else . end) as lead2x1,
       sum(case when have.year+3 = cross.year then cross.x1
                else . end) as lead3x1
 from have
 inner join
 have as cross
 on have.id=cross.id
 group by have.id, have.year;
QUIT;
```

Simulating Lags and Leads

- Output (lag & lead of X1):

ID	YEAR	**X1**	LAG1X1	LAG2X1	LAG3X1	**LEAD1X1**	**LEAD2X1**	**LEAD3X1**
1	1990	**0.3**	.	.	.	**0.45**	**0.4**	**0.25**
1	1991	**0.45**	0.3	.	.	**0.4**	**0.25**	**0.5**
1	1992	**0.4**	0.45	0.3	.	**0.25**	**0.5**	.
1	1993	**0.25**	0.4	0.45	0.3	**0.5**	.	.
1	1994	**0.5**	0.25	0.4	0.45	.	.	.
3	1999	**1.1**	.	.	.	**1.3**	**2.45**	**3.4**
3	2000	**1.3**	1.1	.	.	**2.45**	**3.4**	**2.5**
3	2001	**2.45**	1.3	1.1	.	**3.4**	**2.5**	**4.5**
3	2002	**3.4**	2.45	1.3	1.1	**2.5**	**4.5**	.
3	2003	**2.5**	3.4	2.45	1.3	**4.5**	.	.
3	2004	**4.5**	2.5	3.4	2.45	.	.	.

First Non-Missing Value

- Create example data set

```
data class;
  set sashelp.class;
length Alias $25;
    if Name='James' then Alias='Jim';
      else if Name='Jeffrey' then Alias='Jeff';
      else if Name='Robert' then Alias='Bob';
      else if Name='Thomas' then Alias='Tom';
      else if Name='Barbara' then Alias='Babs';
run;
```

First Non-Missing Value

Name	Sex	Age	Height	Weight	Alias
Alfred	M	14	69.0	112.5	
Alice	F	13	56.5	84.0	
Barbara	F	13	65.3	98.0	Babs
Carol	F	14	62.8	102.5	
Henry	M	14	63.5	102.5	
James	M	12	57.3	83.0	Jim
Jane	F	12	59.8	84.5	
Janet	F	15	62.5	112.5	
Jeffrey	M	13	62.5	84.0	Jeff

First Non-Missing Value

```
PROC SQL;
  create table class2
   as select coalesce(alias,name) as FirstName, sex, age
      from class
      order by FirstName;
QUIT;
```

FirstName	Sex	Age
Alfred	M	14
Alice	F	13
Babs	F	13
Bob	M	12
Carol	F	14

Simulating First. and Last.

- SASHELP.CARS, first 16 observations

Make	Model	Type	Origin	MSRP
Acura	MDX	SUV	Asia	$36,945
Acura	RSX Type S 2dr	Sedan	Asia	$23,820
Acura	TSX 4dr	Sedan	Asia	$26,990
Acura	TL 4dr	Sedan	Asia	$33,195
Acura	3.5 RL 4dr	Sedan	Asia	$43,755
Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	$46,100
Acura	NSX coupe 2dr manual S	Sports	Asia	$89,765
Audi	A4 1.8T 4dr	Sedan	Europe	$25,940
Audi	A41.8T convertible 2dr	Sedan	Europe	$35,940
Audi	A4 3.0 4dr	Sedan	Europe	$31,840
Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	$33,430
Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	$34,480
Audi	A6 3.0 4dr	Sedan	Europe	$36,640
Audi	A6 3.0 Quattro 4dr	Sedan	Europe	$39,640
Audi	A4 3.0 convertible 2dr	Sedan	Europe	$42,490
Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	$44,240

Simulating First. and Last.

```
proc sort data=sashelp.cars out=cars_sorted;
by make msrp;
run;

PROC SQL;
  create table FirstCar(drop=Order) as
   select *, monotonic() as Order
   from cars_sorted
   group by make
   having Order=min(Order);
QUIT;
```

Simulating First. and Last.

- Example output – cheapest model of car per make

Make	Model	Type	Origin	MSRP
Acura	RSX Type S 2dr	Sedan	Asia	$23,820
Audi	A4 1.8T 4dr	Sedan	Europe	$25,940
BMW	325i 4dr	Sedan	Europe	$28,495
Buick	Century Custom 4dr	Sedan	USA	$22,180
Cadillac	CTS VVT 4dr	Sedan	USA	$30,835
Chevrolet	Aveo 4dr	Sedan	USA	$11,690
Chrysler	PT Cruiser 4dr	Sedan	USA	$17,985
Dodge	Neon SE 4dr	Sedan	USA	$13,670
Ford	Focus ZX3 2dr hatch	Sedan	USA	$13,270
GMC	Canyon Z85 SL Regular Cab	Truck	USA	$16,530
Honda	Civic DX 2dr	Sedan	Asia	$13,270

- To get the most expensive model of car per make use **`max(Order)`**

Using PROC SQL to Generate SAS Code

- Dynamically generates a PROC APPEND statement which adds the folders contained in the BDORACLE.table_name table to its equivalent in MS ACCESS library

```
PROC SQL noprint;
 select "proc append base=MSACCESS." || trim(memname) || "
         data=BDORACLE." || trim(memname) || ";run"
 into :command_append separated by ";"
 from sashelp.vtable
 where libname = "MSACCESS";
QUIT;
```

- Resolve the macro variable **&command_append** when you want to execute the SAS code you have created

Accessing an Oracle Database from SAS

- SAS/ACCESS Interface to Oracle
- LIBNAME statement using the ORACLE engine (provides direct and dynamic access to the Oracle data).

```
libname oralib oracle user=scott password=xxxxx
path=V2o7223 preserve_tab-names=yes;
```

- Note: The values for user=, password=, and path= are specific to Oracle. The path= statement is the Oracle alias name that is defined in the Oracle TNSNAMES.ORA file.

Accessing an Oracle Database from SAS

- The SQL Procedure Pass-Through Facility enables you to pass Oracle SQL statements to an Oracle database for processing.

```
PROC SQL;
connect to oracle(user=scott password=tiger
                  path=dark_o8150);
create table test1 as
select * from connection to Oracle
(select * from Dept where rownum <=5);
QUIT;
```

- Can also use SAS/ACCESS Interface to ODBC and SAS/ACCESS Interface to OLE DB

Accessing an SQL Server Database from SAS

- SAS/ACCESS Interface to Microsoft SQL Server
- LIBNAME statement with the SQLSVR engine (provides direct and dynamic, access to the Microsoft SQL Server data).

```
libname mydblib sqlsvr user=testuser password=testpass;
```

- The SQL Procedure Pass-Through Facility enables you to pass MS SQL statements to a SQL Server database for processing.

```
PROC SQL;
connect to SQLSVR as mydb
(datasrc="SQL Server" user=testuser password=testpass);
select * from connection to mydb
(select CUSTOMER, NAME, COUNTRY from CUSTOMERS where
COUNTRY <> 'USA');
QUIT;
```

- Can also use SAS/ACCESS Interface to ODBC and SAS/ACCESS Interface to OLE DB

Acknowledgements and References

Accessing an Oracle Database from SAS on Microsoft Windows

http://support.sas.com/techsup/technote/ts703.pdf

Accessing a Microsoft SQL Server Database from SAS on Microsoft Windows

http://support.sas.com/techsup/technote/ts765.pdf

Tasha Chapman and Lori Carleton, Using SAS® with Oracle®: Writing efficient and accurate SQL

http://www.sascommunity.org/mwiki/images/3/39/Using_SAS_with_Oracle_-_PNWSUG_Paper.pdf

Kirk Paul Lafler, Undocumented and Hard-to-find SQL Features

http://www2.sas.com/proceedings/sugi28/019-28.pdf

SAS Google Groups