

Chapter 1 Overview of the SQL Procedure

1.1	Features of PROC SQL	1-3
1.2	Selecting Columns and Rows	1-6
1.3	Presenting and Summarizing Data	1-17
1.4	Joining Tables.....	1-27

1.1 Features of PROC SQL

Objectives

- Understand SQL procedure syntax.

3

Features of PROC SQL

PROC SQL has the following features:

- is part of Base SAS software
- follows guidelines set by the American National Standards Institute (ANSI)
- can query, combine, create, and update SAS and RDBMS (relational database management system) data
- supports SAS functions such as SUBSTR and MEAN
- supports data set options such as DROP= and RENAME=

4



You must have the appropriate SAS/ACCESS software to query your RDBMS data. You must have appropriate authority to read, create, or update RDBMS objects.

Features of PROC SQL

- The PROC SQL statement does not need to be repeated with each query.
- Each statement is processed individually.
- No PROC PRINT step is needed to view query results.
- No PROC SORT step is needed to order query results.
- No RUN statement is needed.
- Use a QUIT statement to terminate PROC SQL.

5

Features of the SELECT Statement

The features of the SELECT statement include the following:

- selects data that meets certain conditions
- groups data
- specifies an order for the data
- formats the data
- queries 1 to 32 tables

6

Table names and variable names can be 1 to 32 characters in length and are not case-sensitive.

Librefs and filerefs are limited to 8 characters. Starting in SAS[®]9, format and informat names can be up to 32 characters in length.

SELECT Statement Syntax

General form of the SELECT statement:

```
SELECT column-1<, column-2>...  
  FROM table-1|view-1<, table-2|view-2>...  
  <WHERE expression>  
  <GROUP BY column-1<, column-2>...>  
  <HAVING expression>  
  <ORDER BY column-1<, column-2>... <DESC>>;  
QUIT;
```

7

SELECT	specifies the columns to be selected.
FROM	specifies the table to be queried.
WHERE	subsets the data based on a condition.
GROUP BY	classifies the data into groups.
HAVING	subsets groups of data based on a group condition.
ORDER BY	sorts rows of data by the values of specific columns in ascending order by default.

1.2 Selecting Columns and Rows

Objectives

- Display columns directly from a table.
- Display columns calculated from other columns in a query.
- Eliminate duplicate rows in a query.
- Subset the data displayed in a query.

9

Retrieving Data from a Table

If you are familiar with a table, you can specify column names to be printed in the SELECT statement.

Example: Print employee IDs, job codes, and salaries.

```
libname airline 'c:\SQL';  
proc sql;  
    select EmpID, JobCode, Salary  
    from airline.payrollmaster;  
quit;
```

10

Employee IDs, Job Codes, and Salaries

Partial Output

The SAS System		
Emp ID	Job Code	Salary
1919	TA2	\$48,126
1653	ME2	\$49,151
1400	ME1	\$41,677
1350	FA3	\$46,040
1401	TA3	\$54,351
1499	ME3	\$60,235
1101	SCP	\$26,212
1333	PT2	\$124,048
1402	TA2	\$45,661

11

Retrieving Data from a Table

```
proc sql;
  select *
  from airline.payrollmaster;
quit;
```

If you are not familiar with a table, an asterisk in the SELECT statement prints all columns in their originally stored order.

Partial Output

The SAS System					
Emp ID	Gender	Job Code	Salary	DateOfBirth	DateOfHire
1919	M	TA2	\$48,126	16SEP1958	07JUN1985
1653	F	ME2	\$49,151	19OCT1962	12AUG1988
1400	M	ME1	\$41,677	08NOV1965	19OCT1988
1350	F	FA3	\$46,040	04SEP1963	01AUG1988
1401	M	TA3	\$54,351	16DEC1948	21NOV1983
1499	M	ME3	\$60,235	29APR1952	11JUN1978

12

Retrieving Data from a Table

```
proc sql;
  select *
    from airline.payrollmaster(drop=gender);
quit;
```

Partial Output

The SAS System				
Emp ID	Job Code	Salary	DateOfBirth	DateOfHire
1919	TA2	\$48,126	16SEP1958	07JUN1985
1653	ME2	\$49,151	19OCT1962	12AUG1988
1400	ME1	\$41,677	08NOV1965	19OCT1988
1350	FA3	\$46,040	04SEP1963	01AUG1988
1401	TA3	\$54,351	16DEC1948	21NOV1983
1499	ME3	\$60,235	29APR1952	11JUN1978

13

Expressions

Calculate new columns from existing columns, and name the new columns using the AS keyword.

Example: Calculate employee bonuses and age.

```
proc sql;
  select EmpID, JobCode, Salary,
         Salary * .10 as Bonus,
         int((today()-DateOfBirth)/365.25)
         as Age
    from airline.payrollmaster;
quit;
```

14

Employee Bonuses and Age

Partial Output

The SAS System				
Emp ID	Job Code	Salary	Bonus	Age
1919	TA2	\$48,126	4812.64	48
1653	ME2	\$49,151	4915.12	44
1400	ME1	\$41,677	4167.66	41
1350	FA3	\$46,040	4604.04	43
1401	TA3	\$54,351	5435.08	58
1499	ME3	\$60,235	6023.5	55
1101	SCP	\$26,212	2621.22	47
1333	PT2	\$124,048	12404.84	48
1402	TA2	\$45,661	4566.1	46
1479	TA3	\$54,299	5429.9	40

15

Eliminating Duplicate Rows

Use the DISTINCT keyword to eliminate duplicate rows in query results.

Example: Determine the international flights that were flown during the month.

```
proc sql;
  select distinct FlightNumber,
    Destination
  from airline.internationalflights;
quit;
```

16

Eliminating Duplicate Rows

Output

The SAS System	
FlightNumber	Destination
132	YYZ
182	YYZ
219	LHR
271	CDG
387	CPH
622	FRA
821	LHR

17

Subsetting with the WHERE Clause

Use a WHERE clause to specify a condition that the data must satisfy before being selected.

Example: Display all employees that earn more than \$112,000.

```
proc sql;  
  select EmpID, JobCode, Salary  
  from airline.payrollmaster  
  where Salary > 112000;  
quit;
```

18

Subsetting with the WHERE Clause

Output

The SAS System		
Emp ID	Job Code	Salary
1333	PT2	\$124,048
1404	PT2	\$127,926
1118	PT3	\$155,931
1410	PT2	\$118,559
1777	PT3	\$153,482
1106	PT2	\$125,485
1442	PT2	\$118,350
1478	PT2	\$117,884
1890	PT2	\$120,254
1107	PT2	\$125,968
1830	PT2	\$118,259
1928	PT2	\$125,801

19

Subsetting with the WHERE Clause

You can use all common comparison operators in a WHERE clause.

Mnemonic	Symbol	Definition
LT	<	Less than
GT	>	Greater than
EQ	=	Equal to
LE	<=	Less than or equal to
GE	>=	Greater than or equal to
NE	≠	Not equal to (EBCDIC)
	^=	Not equal to (ASCII)

20

Subsetting with the WHERE Clause

You can use the IN operator to compare a value to a list of values. If the value matches at least one in the list, the expression is true; otherwise, the expression is false.

```
where JobCategory in ('PT','NA','FA')
```

```
where DayOfWeek in (2,4,6)
```

21

Subsetting with the WHERE Clause

You can specify multiple expressions in a WHERE clause by using logical operators.

Mnemonic	Symbol	Definition
OR		or, either
AND	&	and, both
NOT	¬	not, negation EBCDIC
NOT	^	not, negation ASCII

22

Subsetting with the WHERE Clause

Use either **CONTAINS** or **?** to select rows that include the substring specified.

```
where word ? 'LAM'
```

(BLAME, LAMENT, and BEDLAM are selected.)

Use either **IS NULL** or **IS MISSING** to select rows with missing values.

```
where FlightNumber is missing
```

23

Subsetting with the WHERE Clause

Use **BETWEEN-AND** to select rows containing ranges of values, inclusively.

```
where Date between '01mar2000'd  
and '07mar2000'd
```

```
where Salary between 70000 and 80000
```

24

Subsetting with the WHERE Clause

Use **LIKE** to select rows by comparing character values to specified patterns.

A % sign replaces any number of characters.

```
where LastName like 'H%'
```

(H plus any characters; for example, HENRY and HAMM.)

A single underscore ('_') replaces individual characters.

```
where JobCode like '__1'
```

(Captures any two characters and 1, for example, 'FA1'.)

25

Subsetting with Calculated Values

Example: Display only the flights where the total number of passengers was fewer than 100 people.

```
proc sql;
  select FlightNumber,Date,Destination,
         Boarded + Transferred + Nonrevenue
         as Total
  from airline.marchflights
  where Total < 100;
```

Partial Log

```
ERROR: The following columns were not
found in the contributing tables: Total.
```

26

Subsetting with Calculated Values

One solution is to repeat the calculation in the WHERE clause.

```
proc sql;
  select FlightNumber, Date, Destination,
         Boarded+Transferred+Nonrevenue
         as Total
  from airline.marchflights
  where Boarded+Transferred+Nonrevenue < 100;
```

27

Subsetting with Calculated Values

A more efficient method is to use the CALCULATED keyword to refer to already calculated columns in the SELECT clause.

```
proc sql;
  select FlightNumber, Date, Destination,
         Boarded + Transferred + Nonrevenue
         as Total
  from airline.marchflights
  where calculated Total < 100;
```

28

Subsetting with Calculated Values

Partial Output

The SAS System			
FlightNumber	Date	Destination	Total
982	01MAR2000	DFW	70
416	01MAR2000	WAS	93
829	01MAR2000	WAS	96
416	02MAR2000	WAS	90
302	02MAR2000	WAS	93

1.3 Presenting and Summarizing Data

Objectives

- Order the data displayed in a query.
- Use SAS formats, labels, and titles to enhance query output.
- Use functions to summarize data in a query.
- Group data for aggregate functions.

31

Ordering Data

The ORDER BY clause has the following functionality:

- sorts on any column or expression (display or nondisplay) in ascending order by default
- can sort data in descending order by following the column name with the DESC keyword
- uses a column name or a number that represents the position of an item in the SELECT list
- can sort on multiple columns

32

Ordering Data

```
proc sql;  
  select EmpID, JobCode, Salary  
  from airline.payrollmaster  
  where JobCode contains 'NA'  
  order by Salary desc;
```

33

Ordering Data

Output

The SAS System		
Emp ID	Job Code	Salary
1352	NA2	\$75,317
1417	NA2	\$73,178
1935	NA2	\$71,513
1839	NA1	\$60,806
1443	NA1	\$59,184
1332	NA1	\$59,049
1269	NA1	\$58,366
1111	NA1	\$56,820

34

Enhancing Query Output

You can use SAS titles and footnotes enhance your output.

You can also use SAS formats and labels to customize PROC SQL output. After the column name in the SELECT list, you specify the following:

- LABEL= option to alter the column heading
- FORMAT= option to alter the appearance of the values in that column.

35

Enhancing Query Output

Example: Enhance the report. Display the navigators and their salaries.

```
proc sql;
title 'Navigator Salaries';
  select EmpID label='Employee Identifier',
         JobCode label='Job Code',
         Salary label='Annual Salary'
         format=dollar12.2
  from airline.payrollmaster
  where JobCode contains 'NA'
  order by Salary desc;
```

36

Enhanced Query Output

Output

Navigator Salaries		
Employee Identifier	Job Code	Annual Salary
1352	NA2	\$75,317.20
1417	NA2	\$73,178.00
1935	NA2	\$71,513.40
1839	NA1	\$60,806.20
1443	NA1	\$59,183.60
1332	NA1	\$59,049.20
1269	NA1	\$58,366.00
1111	NA1	\$56,820.40

37

Summary Functions

Example: Find the total number of passengers for each flight in March.

```
proc sql;
  select Date, FlightNumber, Boarded,
         Transferred, Nonrevenue,
         sum(Boarded, Transferred, Nonrevenue)
         as Total
  from airline.marchflights;
quit;
```

This calculation is performed *across columns* for each row.

38

Summary Functions

Partial Output

The SAS System					
Date	FlightNumber	Boarded	Transferred	Nonrevenue	Total
30MAR2000	183	86	5	4	95
30MAR2000	271	164	15	6	185
30MAR2000	921	101	20	4	125
30MAR2000	302	54	14	3	71
30MAR2000	431	159	14	4	177
30MAR2000	308	121	18	6	145
31MAR2000	182	91	7	6	104
31MAR2000	114	183	14	4	201
31MAR2000	202	111	12	3	126

39

Summary Functions

If you specify only one column name in a summary function, the statistic is calculated down the column.

Example: Determine the average salary for the company.

```
proc sql;
  select avg(Salary) as MeanSalary
  from airline.payrollmaster;
```

Output

The SAS System	
MeanSalary	
	54079.65

40

Summary Functions

The following are selected functions:

AVG, MEAN	mean or average value
COUNT, FREQ, N	number of nonmissing values
MAX	largest value
MIN	smallest value
NMISS	number of missing values
STD	standard deviation
SUM	sum of values
VAR	variance

41

Summary Functions

Example: Add the JobCode column to the summarized query.

```
proc sql;  
  select JobCode, avg(Salary) as Average  
  from airline.payrollmaster;
```

42

Summary Functions

Partial Output

The SAS System	
Job Code	Average
TA2	54079.65
ME2	54079.65
ME1	54079.65
FA3	54079.65
TA3	54079.65
ME3	54079.65
SCP	54079.65
PT2	54079.65
TA2	54079.65
TA3	54079.65
ME1	54079.65

43

Grouping Data

You can use the GROUP BY clause to

- classify the data into groups based on the values of one or more columns
- calculate statistics for each unique value of the grouping columns.

44

Grouping Data

Example: Display the average salary and number of employees for each job code.

```
proc sql;  
  select JobCode, avg(Salary) as  
         Average format=dollar11.2,  
         count(*) as Employees  
  from airline.payrollmaster  
  group by JobCode;
```

45

Grouping Data

Partial Output

The SAS System		
Job Code	Average	Employees
BCK	\$36,111.91	9
FA1	\$32,255.11	11
FA2	\$39,181.63	16
FA3	\$46,107.40	7
ME1	\$39,900.35	8
ME2	\$49,807.60	14
ME3	\$59,375.00	7
NA1	\$58,845.08	5
NA2	\$73,336.20	3
PT1	\$95,071.20	8

46

Selecting Groups of Data with the HAVING Clause

The WHERE clause selects data based on values for individual rows. To select entire groups of data, use the HAVING clause.

Example: Display all job codes with an average salary of more than the company average salary of \$54,079.

```
proc sql;
  select JobCode, avg(Salary) as Average
         format=dollar11.2
  from airline.payrollmaster
  group by JobCode
  having avg(salary) > 54079;
```

47

To avoid using hard-coded constants, use a subquery to return the company average salary. A subquery, or inner query, is a query-expression that is nested as part of another query-expression. The subquery executes first, returning its results to the outer query.

```
select JobCode, avg(Salary) as Average format=dollar11.2
  from airline.payrollmaster
  group by JobCode
  having avg(Salary) > (select avg(Salary)
                       from airline.payrollmaster);
```

Selecting Groups of Data with the HAVING Clause

Output

The SAS System	
Job Code	Average
ME3	\$59,375.00
NA1	\$58,845.08
NA2	\$73,336.20
PT1	\$95,071.20
PT2	\$122,253.60
PT3	\$154,706.30

1.4 Joining Tables

Objectives

- Join two tables.
- Use a table alias.
- Create a table from the join query results.

51

Combining Data from Multiple Tables

Joins combine tables horizontally (side by side).

Table A		Table B
---------	--	---------

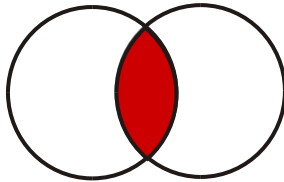
PROC SQL supports both inner and outer joins. We will look at inner joins.

51

Inner Joins

Inner joins have the following characteristics:

- return only matching rows
- allow a maximum of 32 tables to be joined at the same time



53

Inner Joins

An *inner join* returns only the subset of rows from the first table that matches rows from the second table. You can specify the columns that you want to be compared for matching values in a WHERE clause.

General form of an inner join:

```
SELECT column-1, column-2, ...
FROM table-1, table-2, ...
WHERE join-condition(s)
       <AND other subsetting conditions>
       <other clauses>;
```

53

Without a WHERE clause, the join will return a *Cartesian product* of the tables where each row from the first table is combined with every row from the second table.

Inner Joins

Example: Create a table that contains the names, job codes, and ages of all New York employees.

- Employee names are found in the `airline.staffmaster` table.
- Employee job codes and birth dates are found in the `airline.payrollmaster` table.

55

Inner Joins

```
proc sql;
  title 'New York Employees';
  select FirstName, LastName, JobCode,
         int((today()-DateOfBirth)/365.25)
         as Age
  from airline.payrollmaster as p,
       airline.staffmaster as s
  where p.EmpID=s.EmpID and State='NY'
  order by JobCode;
quit;
```

56

An *alias* is a table nickname. You can assign an alias to a table by following the table name in the FROM clause with the AS keyword and a nickname for the table. Then use the alias in other clauses of the QUERY statement.

Inner Joins

Partial Output

New York Employees			
FirstName	LastName	Job Code	Age
RUSSELL	LONG	BCK	37
LEVI	GORDON	BCK	50
JAMES	PEARSON	BCK	49
NATHAN	JONES	BCK	42
THOMAS	BURNETTE	BCK	41
RICHARD	VANDEUSEN	BCK	48
JOHN	MARKS	BCK	42
DEBORAH	WOOD	FA1	38
LESLIE	JONES	FA1	41
ANNE	PARKER	FA1	44
DIANA	FIELDS	FA1	51
CASEY	RICHARDS	FA1	39

57

Creating a Table From Query Results

The **CREATE TABLE *table-name* AS** statement stores the results of the query into a table.

```
proc sql;
  create table airline.newyork as
    select FirstName, LastName, JobCode,
           int((today()-DateOfBirth)/365.25)
           as Age
    from airline.payrollmaster as p,
         airline.staffmaster as s
    where p.EmpID=s.EmpID and State='NY'
    order by JobCode;
quit;
```

58

The CREATE TABLE statement does not generate output.

Creating a Table From Query Results

```
proc print data=airline.newyork noobs;
  title 'New York Employees';
run;
```

Partial Output

New York Employees			
FirstName	LastName	Job Code	Age
RUSSELL	LONG	BCK	37
LEVI	GORDON	BCK	50
JAMES	PEARSON	BCK	49
NATHAN	JONES	BCK	42
THOMAS	BURNETTE	BCK	41
RICHARD	VANDEUSEN	BCK	48
JOHN	MARKS	BCK	42
DEBORAH	WOOD	FA1	38
LESLIE	JONES	FA1	41

59

Where to Go Next

SQL Processing with the SAS® System – 2 days

PROC SQL: Beyond the Basics Using SAS®
by Kirk Paul Laffler

The Essential PROC SQL Handbook for SAS® Users
by Katherine Prairie

SQL Self-Paced eLearning:
<http://www.sas.com/apps/elearning/>

58