

: Conditional Logic, Macro Variables and Subqueries.

By
Christopher Zogby

Outline

- Introduction to SQL
 - ◆ Basic SQL syntax
- Conditional Logic
 - ◆ Syntax/Examples
- Macro Variables
 - ◆ Syntax/Examples
- Subqueries
 - ◆ Syntax/Examples

Basic SQL Syntax

- What is SQL?
- SELECT, FROM, WHERE statements
- CREATE/DROP/ALTER statements
- ORDER BY, GROUP BY clauses

What is SQL?

- SQL = structured query language
- An ANSI database language
 - ◆ ANSI = American National Standards Institute
- SQL functionality provided as a SAS procedure (i.e. PROC SQL;)

SELECT, FROM, WHERE statements

- **SELECT**
 - ◆ comma delimited variable list, or
 - ◆ ‘*’, for all variables in table or view
- **FROM**
 - ◆ table name
 - ◆ view name
- **Where**
 - ◆ clause that determines which rows pass to results set.

SELECT-FROM

Example

```

proc sql;

title 'Select entire contents of table, SASHELP.CLASS.';

SELECT *
FROM sashelp.class;

title;

```

Select entire contents of table, SASHELP.CLASS.

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

SELECT-FROM-WHERE

EXAMPLE

```
title 'Select girls with first names that begin with '  
      'J from table, SASHELP.CLASS.:'  
  
SELECT name,  
       sex  
FROM sashelp.class  
WHERE name like 'J%' and  
       sex='F'  
;  
title;  
  
quit;
```

Select girls with first names that begin with J from table, SASHELP.CLASS.

Name	Sex
Jane	F
Janet	F
Joyce	F
Judy	F

CREATE/DROP/ALTER statements

- CREATE
 - ◆ table/view/index
- DROP
 - ◆ table/view/index
- ALTER
 - ◆ table
 - ☞ add (column-definition, constraint-clause)
 - ☞ modify (column-definition)
 - ☞ drop (column-definition, constraint clause)

CREATE-DROP-ALTER

Example

```

proc sql;

libname devl 'c:\development';

create table devl.class as          /* <-- CREATE TABLE */
select *
from sashelp.class
;

drop table devl.class;             /* <-- DROP TABLE */

libname devl;

create table o_negative_donors     /* <-- CREATE TABLE */
(donor_name  varchar(40),
 blood_type  varchar(3),
 rh_factor   char(1)
)
;

alter table o_negative_donors      /* <-- ALTER TABLE          */
add donor_comments varchar(1000), /* <-- ADD COLUMN, CONSTRAINT */
    constraint only_o_neg check(blood_type = 'O' and
                                rh_factor='-')
);

drop table o_negative_donors;      /* <-- DROP TABLE */

quit;

```

ORDER BY, GROUP BY clauses

■ GROUP BY

- ◆ Used when summarizing by one or more class variable
- ◆ comma delimited

■ Order BY

- ◆ Used when sorting resulting data
- ◆ comma delimited list
- ◆ ascending or descending order

GROUP BY-ORDER BY

Example

```
proc sql;
title 'Student frequency by sex.';

select  sex,
        count(*) as count
from sashelp.class
group by sex
order by count desc
;

title;

quit;
```

Student frequency by sex.

Sex	count
M	10
F	9

Outline

- Introduction to SQL
 - ◆ Basic SQL syntax
- Conditional Logic
 - ◆ Syntax/Examples

Conditional Logic

- Conditionally assign values to a variable.
- Valid in SELECT/UPDATE/INSERT statement.
- CASE expression

CASE expression syntax

```

CASE <case-operand>
    WHEN when-condition THEN result-expression

    <WHEN when-condition THEN result-expression>...

    <ELSE result-expression>

END
  
```

CASE expression's rules of operation.

- Need at least one WHEN-THEN clause
- Need END to terminate the expression
- CASE operand present
 - ◆ WHEN condition & operand compared for equality
 - ☞ operand=WHEN condition, adjoining THEN's result-expression executes
 - no further WHEN-THEN statements are evaluated.
 - ☞ operand ^= WHEN condition, following WHEN condition(s) evaluated.

Rules of operation (continued).

- CASE operand omitted
 - ◆ WHEN condition evaluated as Boolean value
 - ☞ WHEN condition returns non-missing, non-zero result, adjoining THEN's result-expression executes
 - no further WHEN-THEN statements are evaluated.
 - ☞ If operand \neq WHEN condition, following WHEN condition(s) evaluated.
- No WHEN conditions true, ELSE's result-expression or (without ELSE) null returned

CASE Expression: no operand.

Example

```

proc sql;

title "CASE expression, without operand.";

select weight,
       case
         when weight = .           then ''
         when . < weight < 80     then 'Light'
         when 80 <= weight <= 110 then 'Medium'
         else                       'Heavy'
       end as weight_class

from sashelp.class;

title;

```

CASE expression, without operand.

Weight	weight_class
112.5	Heavy
84	Medium
98	Medium
102.5	Medium
102.5	Medium
83	Medium
84.5	Medium
112.5	Heavy
84	Medium
99.5	Medium
50.5	Light
90	Medium
77	Light
112	Heavy
150	Heavy
128	Heavy
133	Heavy
85	Medium
112	Heavy

CASE Expression: operand.

Example

```
title "CASE expression, with operand.";

select sex,
       case sex
         when 'M' then 'BOY'
         when 'F' then 'GIRL'
       end as boy_or_girl
from sashelp.class;

title;

quit;
```

CASE expression, with operand.

Sex boy_or_girl	

M	BOY
F	GIRL
F	GIRL
F	GIRL
M	BOY
M	BOY
F	GIRL
F	GIRL
M	BOY
M	BOY
F	GIRL
F	GIRL
F	GIRL
F	GIRL
M	BOY
M	BOY
M	BOY
M	BOY
M	BOY

Outline

- Introduction to SQL
 - ◆ Basic SQL syntax
- Conditional Logic
 - ◆ Syntax/Examples
- Macro Variables
 - ◆ Syntax/Examples

Macro Variables

- Outer query's SELECT statement
- Not valid when a table or view is created
- Can create macro variable for each queried column
- Macro variable can store single or multiple rows from queried column
- Macro variable created using INTO clause

INTO clause syntax

INTO *:macro-variable-specification*
<, *:macro-variable-specification*>...

- *:macro-variable-specification* is one of the following:
:macro-variable <SEPARATED BY 'character' <NOTRIM>>;

:macro-variable-1 - :macro-variable-n <NOTRIM>;

Macro Variables: multiple columns, single row.

Example

```

proc sql;

reset noprint;

select round(mean(height),0.01),
       round(mean(weight),0.01)
                                into :avg_height_boys,
                                :avg_weight_boys

from sashelp.class
where sex='M'
;

reset print;

title 'Boys below average weight ('%trim(&avg_weight_boys)
      ') and average height ('%trim(&avg_height_boys)')';

select *
from sashelp.class
where sex='M'                and
      weight < &avg_weight_boys and
      height < &avg_height_boys
;
quit;

```

Boys below average weight (108.95) and average height (63.91).

Name	Sex	Age	Height	Weight
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Thomas	M	11	57.5	85

Macro Variables: variable for each resulting row.

Example

```
proc sql;

reset noprint;

select count(*) into :girls
from sashelp.class
where sex='F'
;
select weight into :gweight1 through :gweight%left(&girls)
from sashelp.class
where sex='F'
;
reset number print;
title "Girls' weight. Third girl's weight=%trim(&gweight3).";
select sex, weight
from sashelp.class
where sex='F' ;
title;
quit;
```

Girls' weight. Third girl's weight=102.5.

Row	Sex	Weight
1	F	84
2	F	98
3	F	102.5
4	F	84.5
5	F	112.5
6	F	50.5
7	F	90
8	F	77
9	F	112

Macro Variables: multiple, delimited rows stored in single variable.

Example

```

/* Create Temp Tables to be dropped */
data temp1 temp2 temp3;
run;

proc sql;

reset noprint;

select trim(libname)||'.'||trim(memname)

        into :temp_files separated by ', '

from dictionary.tables
where libname='WORK' and
      memtype='DATA' and
      memname like 'TEMP%'
;

%put &temp_files=;

drop table &temp_files;

quit;

```

```
169 proc sql;
170
171 /* Macro Variables: Multiple, Delimited Rows Stored in
Single Variable */
172 reset noprint;
173
174 select trim(libname)||'.'||trim(memname)
175
176         into :temp_files separated by ', '
177
178 from dictionary.tables
179 where libname='WORK' and
180        memtype='DATA' and
181        memname like 'TEMP%'
182 ;
183
184 %put &temp_files=;
WORK.TEMP1, WORK.TEMP2, WORK.TEMP3=
185
186 drop table &temp_files;
NOTE: Table WORK.TEMP1 has been dropped.
NOTE: Table WORK.TEMP2 has been dropped.
NOTE: Table WORK.TEMP3 has been dropped.
187
188 quit;
```

Outline

- Introduction to SQL
 - ◆ Basic SQL syntax
- Conditional Logic
 - ◆ Syntax/Examples
- Macro Variables
 - ◆ Syntax/Examples
- Subqueries
 - ◆ Syntax/Examples

Subqueries (SQ)

- Contained within parenthesis.
- SQ is query that execute within context of another query.
- Innermost SQ executes first, followed by next innermost, and so on until outer query is executed.
- Can return single or many rows.
 - ◆ Single row SQ can be evaluated like constant or variable in next outermost query.
 - ◆ Multi-row SQ can be evaluated with IN operator, or treated as a separate table in FROM statement.

Subquery: single row returned in
INSERT statement.

Example

```

proc sql;

insert into avg_growth_monthly
set month      = month(today()),
  year        = year(today()),
  weight_boys = (select avg(weight) from sashelp.class
                 where sex='M'
                ),
  height_boys = (select avg(height) from sashelp.class
                 where sex='M'
                ),
  weight_girls = (select avg(weight) from sashelp.class
                  where sex='F'
                 ),
  height_girls = (select avg(height) from sashelp.class
                  where sex='F'
                 )
;
title "Monthly average height & weight.";
select *
from avg_growth_monthly;
title;

quit;

```

Monthly average height & weight.

month	year	weight_boys	height_boys	weight_girls	height_girls
11	2002	108.95	63.91	90.11111	60.58889

Subquery: multi-row, evaluated by outer select's IN operator

Example

```
proc sql;

create table postmortem_ge10K_payments as
select *
from daily_payments
where amount >= 10000 and
      vet_id in(select veteran_id
                from vet_demographics
                where death_date is not null
                );

quit;
```

Subquery: referenced as a table.

Example

```

proc sql;
title "Students 10% below or above sex-specific average height &
weight.";

select a.*,
       round(b.sex_specific_avg_weight,0.01) as avg_weight,
       round(b.sex_specific_avg_height,0.01) as avg_height,
       case when weight < b.sex_specific_avg_weight then '-'
            else '+'
            end as indicator
from sashelp.class a,
     (select sex,
          avg(weight) as sex_specific_avg_weight,
          avg(height) as sex_specific_avg_height
     from sashelp.class
     group by sex
     ) b
where a.sex=b.sex and
     (
(a.weight/b.sex_specific_avg_weight le .90 and
a.height/b.sex_specific_avg_height le .90
)
or
(a.weight/b.sex_specific_avg_weight ge 1.10 and
a.height/b.sex_specific_avg_height ge 1.10
)
);

quit;

```

Students 10% below or above sex-specific average height & weight.

Name	Sex	Age	Height	Weight	avg_weight	avg_height	indicator
James	M	12	57.3	83	108.95	63.91	-
Joyce	F	11	51.3	50.5	90.11	60.59	-
Philip	M	16	72	150	108.95	63.91	+
Thomas	M	11	57.5	85	108.95	63.91	-

Conclusions

- SAS's SQL extends beyond SELECT, FROM and WHERE statements.
 - ◆ Conditional logic provides flexibility in variable creation.
 - ◆ Macro variable creation allows some automation of SQL code.
 - ◆ Subqueries make SQL a compact, powerful & sometimes novel coding solution.

Questions or Comments

E-mail: czogby@erols.com