TATA CONSULTANCY SERVICES
Experience certainty.

SAS® Training

Basic Proc SQL

What is SQL?

SQL stands for **S**tructured **Q**uery **L**anguage

Used for **retrieving** and **updating** data in relational tables and databases

Word Relation is nothing but a mathematical concept of set.

Relations are represented physically as two-dimensional tables that are arranged in rows and columns

Topics covered…

- Basic syntax of PROC SQL
- Creating tables
- Joining tables
- Group functions
- CASE / WHEN logic

What can you do with SQL?

Create a table	Data step
Sort observations	Proc Sort
List records	Proc Print
Join tables	Data step merge
Summarize data	Proc Freq, Proc Means, Proc Report, Proc Tabulate, etc.

Basic SQL Query

```
Proc SQL;

Select pt ,age , sex

From dm;

Quit;
```

Output

PT	AGE	SEX
4005	22	Male
4007	21	Male
4008	38	Female
5001	40	Female
5002	48	Male
5003	47	Female
5004	48	Female
5005	46	Male
5006	24	Male
5007	48	Female
5008	22	Male

Where Clause

```
Proc SQL;

Select pt, age, sex

From dm

Where sex= 'Female';

Quit;
```

Output

PT	AGE	SEX
1006	55	Female
1007	55	Female
1008	56	Female
2001	53	Female
2005	55	Female
3001	45	Female
3008	54	Female
4008	38	Female
5001	40	Female
5003	47	Female
5004	48	Female
5007	48	Female

Order By Clause

```
Proc SQL;
Select pt, age, sex
From dm
Where sex = 'Female'
Order by age;
Quit;
```

Order by Output

PT	AGE	SEX
4008	38	Female
5001	40	Female
3001	45	Female
5003	47	Female
5007	48	Female
5004	48	Female
2001	53	Female
3008	54	Female
1006	55	Female
1007	55	Female
2005	55	Female
1008	56	Female

Basic SQL Query

```
Proc SQL;

Select pt, age, sex

From dm

Where sex = 'Female';

Quit;
```

Select the variables (columns) you want to keep

Identify the table(s) the data comes from

Indicate which observations (rows) you want to select

Basic SQL Query

```
Proc SQL;

Select pt, age, sex

From dm

Where sex = 'Female';

Quit;
```

Separate variables with commas

End with QUIT; not RUN;

Only one semi-colon at the end of the query

Ordering the SELECT Statement

When you construct a SELECT statement, you must specify the clauses in the following order:

SELECT:- extracts data from a database

FROM:-Identify the table(s) the data comes from

WHERE:- The WHERE clause is used to filter records.

GROUP BY:- Aggregate functions often need an added GROUP BY statement.

HAVING:- HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions

ORDER BY:- ORDER BY keyword is used to sort the result-set.

Note: Only the SELECT and FROM clauses are required.

Select *

- Use an asterisk to select all the available columns in a table

```
Proc SQL;

Select *

From dm;

Quit;
```

Output for all variable

```
STUDY           DCMNAME         DCMSUBNM        SUBSETSN        DOCNUM          INVSITE         INV
PT              ACCESSTS        LOGINTS         LSTCHGTS        LOCKFLAG        CPEVENT         DCMDATE
                                                SUBEVENT_                       QUALIFYING_     QUALIFYING_
DCMTIME         REPEATSN        ACTEVENT        NUMBER          VISIT_NUMBER    VALUE           QUESTION
                LABRANGE_       LAB_ASSIGNMENT_
LAB             SUBSET_NUMBER   TYPE_CODE          LAB_ID          BRTHDT          SEX             ETHNIC
RACE            OTHSP           AGE
-----------------------------------------------------------------------------------------------------------------
SNO2            DM              DM              1               R10750801       N30_SITE02      INVLAKE
5008            04-NOV-14       04-NOV-14       04-NOV-14       Y               SCREENING
                1               10              0               10              001             18001
                                                                   19921210        Male            Not Hispanic
                                                                                                   or Latino
White                           22
```

Renaming variables

- Use AS to rename a variable or name a newly created variable

```
Proc SQL;
Select pt , age, sex as gender
From dm;
Quit;
```

Renaming variable output

PT	AGE	gender
4005	22	Male
4007	21	Male
4008	38	Female
5001	40	Female
5002	48	Male
5003	47	Female
5004	48	Female
5005	46	Male
5006	24	Male
5007	48	Female
5008	22	Male

Labels and Formats

- Labels and Formats can be applied to variables in the SELECT clause

```
Proc SQL;

Select pt label='Subject ID',
       bthdat format=date9.

From dm;

Quit;
```

Labels and Formats output

Subject ID	bthdat
4005	12MAY1992
4007	21AUG1993
4008	21JUN1976
5001	07JUL1974
5002	14SEP1966
5003	21OCT1967
5004	07FEB1967
5005	01DEC1968
5006	31MAY1990
5007	18JAN1967
5008	10DEC1992

Function

```
Proc SQL;
Select  avg(age) as ageavg
label="AVERAGE AGE"
From dm
Quit;
```

Function

```
Proc SQL;
Select sex , avg(age) as ageavg
label="AVERAGE AGE'
From dm
Group by sex;
Quit;
```

Output for

```
 AVERAGE
     AGE
---------
39.12245
```

SEX	AVERAGE AGE
Female	49.5
Male	35.75676

Functions

Summary Function	Description
AVG, MEAN	Average or mean of values
COUNT, FREQ, N	Aggregate number of non-missing values
CSS	Corrected sum of squares
CV	Coefficient of variation
MAX	Largest value
MIN	Smallest value
NMISS	Number of missing values
PRT	Probability of a greater absolute value of Student's t
RANGE	Difference between the largest and smallest values
STD	Standard deviation
STDERR	Standard error of the mean
SUM	Sum of values
SUMWGT	Sum of the weight variable values, which is 1
T	Testing the hypothesis that the population mean is zero
USS	Uncorrected sum of squares
VAR	Variance

Function

```
Proc SQL;
Select pt, sex ,age
From dm
Group by sex
Having age=min(age);
Quit;
```

Output

PT	SEX	AGE
4008	Female	38
2003	Male	19

SQL BASIC

DISTINCT statement is used to return only distinct (different) Values.

The COUNT() function returns the number of rows that matches a specified criteria.

Function

```
Proc SQL;
Select count(usubjid) as totcnt
From adsl
Quit;
```

```
  totcnt
--------
      49
```

Function

```
Proc SQL;
Select trt01p, count(distinct
usubjid) as count
From adsl
Group by trt01p;
Quit;
```

Output

TRT01P	count
N91115 10 mg	8
N91115 250 mg	17
N91115 50 mg	16
N91115 500 mg	8

N91115-1H-01 (SNO-2) Page X of Y

Table 14.1.2 Demographic Characteristics Summary (All Randomized Subjects)

Demographic Characteristics	Category/ Statistics	N91115 10 mg (N=XX)	N91115 50 mg (N=XX)	N91115 250 mg (N=XX)	Placebo (N=XX)	Total (N=XX)
Gender	Male, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Female, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Race	White, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Black or African American, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Asian, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Native American or Alaska Native, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Native Hawaiian or other Pacific Islander, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Other, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Ethnicity	Hispanic or Latino, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
	Not Hispanic or Latino, n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Age (years)	n	xx	xx	xx	xx	xx
	Mean	xx.x	xx.x	xx.x	xx.x	xx.x
	SD	x.xx	x.xx	x.xx	x.xx	x.xx
	Median	xx.x	xx.x	xx.x	xx.x	xx.x
	Minimum	xx	xx	xx	xx	xx
	Maximum	xx	xx	xx	xx	xx

SD - Standard Deviation
Data Source: dataset_name Filename: program_name.sas
Programmer: programmer_initials ddmonyyyyhh:mm SAS 9.2

Function

```
Proc SQL;
Select count(distinct usubjid) as
count1 into: count1
From adsl
Where trt01p="N91115 10 mg";
Quit;
```

Creating Datasets

```
Proc SQL;
Create Table Newdata as
Select pt, age, sex
From dm
Where sex = 'Female';
Quit;
```

Joining Tables

- An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

  Before we continue with examples, we will list the types of the different SQL JOINs you can use:

- **`INNER JOIN`**: Returns all rows when there is at least one match in BOTH tables
- **`LEFT JOIN`**: Return all rows from the left table, and the matched rows from the right table
- **`RIGHT JOIN`**: Return all rows from the right table, and the matched rows from the left table
- **`FULL JOIN`**: Return all rows when there is a match in ONE of the tables

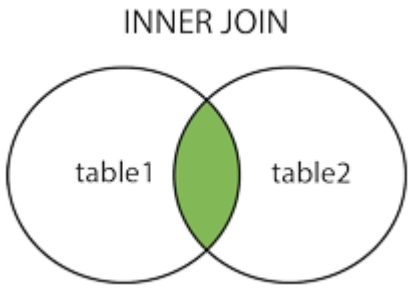
INNER Join

An INNER JOIN combines and displays only the rows from the first table that match rows form the second table

Synatax:

```
Proc sql;
SELECT column_name(s)
FROM table1
INNER JOIN
table2
ON table1.column_name=table2.column_name;
Quit;
```

INNER Join



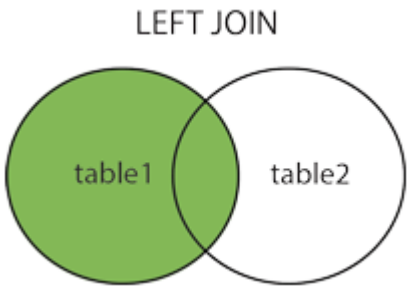
LEFT Join

LEFT JOIN returns all rows from the left table(table 1) , with the matching rows in the right table(table 2).

Synatax:

```
Proc sql;
SELECT column_name(s)
FROM table1
LEFT JOIN /LEFT OUTER JOIN
table2
ON table1.column_name=table2.column_name;
Quit;
```

LEFT Join



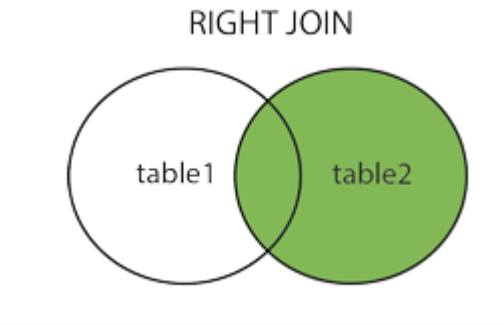
RIGHT Join

RIGHT JOIN returns all rows from the right table(table 2) , with the matching rows in the left table(table 1).

Synatax:

```
Proc sql;
SELECT column_name(s)
FROM table1
RIGHT JOIN /RIGHT OUTER JOIN
table2
ON table1.column_name=table2.column_name;
Quit;
```

RIGHT Join



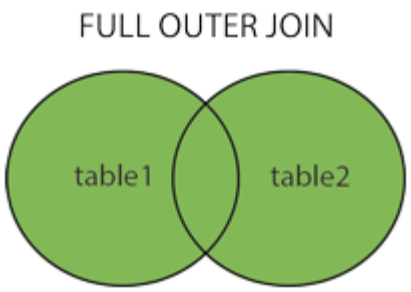
FULL Join

FULL OUTER JOIN returns all rows from the left table(table 1) and from the right table (table 2).

Synatax:

```
Proc sql;
SELECT column_name(s)
FROM table1
FULL OUTER JOIN
table2
ON table1.column_name=table2.column_name;
Quit;
```

FULL Join



Tests Table

Date	Student_ID	Subject	Session	Score
01/25/10	A12	Math	A	96
01/25/10	B34	Math	A	92
01/25/10	C56	Math	A	68
01/25/10	D75	Math	A	79
03/26/10	B34	Science	A	96
03/26/10	C56	Science	A	82
04/23/10	A12	Reading	A	84
04/23/10	B34	Reading	A	94
04/23/10	C56	Reading	A	78
04/23/10	D75	Reading	A	81
05/22/10	A12	Math	B	92
05/22/10	B34	Math	B	94
05/22/10	C56	Math	B	72
05/22/10	D75	Math	B	81
04/15/10	B34	Science	B	94
04/15/10	C56	Science	B	84
06/01/10	A12	Reading	B	88
06/01/10	B34	Reading	B	96
06/01/10	C56	Reading	B	82
06/01/10	D75	Reading	B	79

Convert “Scores” to letter grades

CASE/WHEN

```
data NewGrade; set Tests;
 length Test_Grade $ 4;
 if 70 le Score le 79 then Test_Grade = 'C';
 else if 80 le Score le 89 then Test_Grade = 'B';
 else if 90 le Score le 100 then Test_Grade = 'A';
 else Test_Grade = 'Fail';
run;
```

CASE/WHEN

```
Case
  when 70 le Score le 79 then 'C'
  when 80 le Score le 89 then 'B'
  when 90 le Score le 100 then 'A'
  else 'Fail'
  end
as Test_Grade
```

CASE/WHEN

WHEN works like "ELSE IF"

```
Case
  when 70 le Score le 79 then 'C'
  when 80 le Score le 89 then 'B'
  when 90 le Score le 100 then 'A'
  else 'Fail'
  end
as Test_Grade
```

CASE/WHEN

```
Case
  when 70 le Score le 79 then 'C'
  when 80 le Score le 89 then 'B'
  when 90 le Score le 100 then 'A'
  else 'Fail'
  end
as Test_Grade
```

Conclude with an "END" expression

CASE/WHEN

```
Proc SQL;
Select Date, Student_ID, Subject,
          Session, Score,
          Case
             when 70 le Score le 79 then 'C'
             when 80 le Score le 89 then 'B'
             when 90 le Score le 100 then 'A'
             else 'Fail'
          end
          as Test_Grade
From Tests;
Quit;
```


47

Results

Date	Student_ID	Subject	Session	Score	Test_Grade
01/25/10	A12	Math	A	96	A
01/25/10	B34	Math	A	92	A
01/25/10	C56	Math	A	68	Fail
01/25/10	D75	Math	A	79	C
03/26/10	B34	Science	A	96	A
03/26/10	C56	Science	A	82	B
04/23/10	A12	Reading	A	84	B
04/23/10	B34	Reading	A	94	A
04/23/10	C56	Reading	A	78	C
04/23/10	D75	Reading	A	81	B
05/22/10	A12	Math	B	92	A
05/22/10	B34	Math	B	94	A
05/22/10	C56	Math	B	72	C
05/22/10	D75	Math	B	81	B
04/15/10	B34	Science	B	94	A
04/15/10	C56	Science	B	84	B
06/01/10	A12	Reading	B	88	B
06/01/10	B34	Reading	B	96	A
06/01/10	C56	Reading	B	82	B
06/01/10	D75	Reading	B	79	C

TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Thank You

IT Services
Business Solutions
Outsourcing