Statistical Analysis System: Class 30
Dated: 23/06/2018

Views: Virtual table, it doesn't store data physically. It stores a select query. Benefits of Views:--

- Saves space & money
- Brings data on fly (at runtime)
- Data security

Describe: is the keyword to check metadata of a table through Sql. It is similar to Proc Contents in base sas.

Code	Log / Output
Proc sql; Describe table sasuser.admit; Quit;	NOTE: SQL table SASUSER.ADMIT was created like: create table SASUSER.ADMIT(bufsize=8192) (ID char(4), Name char(14), Sex char(1), Age num, Date num, Height num, Weight num, ActLevel char(4), Fee num format=7.2); 3 quit;
Proc sql Create view a as select * from sasuser.admit; Quit;	64 create view a as select* from sasuser.admit; OTE: SQL view WORK.A has been defined. 65 quit; OTE: PROCEDURE SQL used (Total process time):
Proc sql; Describe view a; Quit;	10 proc sql; 11 describe view a; NOTE: SQL view WORK.A is defined as: select * from SASUSER.ADMIT; 12 quit; NOTE: PROCEDURE SQL used (Total process time): real time 0.03 seconds cpu time 0.03 seconds
Data c; Set a; Run;	Gets the content from view 'a' as output in the dataset 'c'.

Ways to create tables:

1. **Column definition way**: creating table below, it would be empty;

```
proc sql;
create table temp (name char,sex char(2), age num 'age of subject' informat=8. format=8.);
quit;
```

Log display:

```
23   create table temp (name char,sex char(2),age num 'ag
23 ! format=8.);
NOTE: Table WORK.TEMP created, with 0 rows and 3 columns.
24   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.07 seconds
      cpu time            0.07 seconds
```

2. Using "Like": Creating tables, it would be empty;

```
Proc sql;
Create table megh like sasuser.admit;
quit;
```

Log display:

```
25   proc sql;
26   create table megh like sasuser.admit;
NOTE: Table WORK.MEGH created, with 0 rows and 9 columns
27   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds
```

3. Creating tables using select

```
Proc sql;
Create table megh as select *from sasuser.admit;
quit;
```

Log display:

```
NOTE: Table WORK.MEGH created, with 21 rows and 9 columns

30   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.09 seconds
      cpu time            0.06 seconds
```

Ways to insert rows:

1. **Inserting rows using "set"**

```
Proc sql;
Create table temp as select * from sasuser.admit (keep=name age sex);
quit;
proc sql;
insert into temp
set name='Amit',
    age=29,
    sex='M'

set name='Preeti',
    age=28
    sex='F'
;
```

quit;

Log display:

```
52    proc sql;
53    insert into temp1
54    set name='amit',age=29,sex='M'
55    set name='preeti', age= 28, sex='F'
56    ;
NOTE: 2 rows were inserted into WORK.TEMP1.

57    quit;
NOTE: PROCEDURE SQL used (Total process time
      real time           0.03 seconds
```

2. Inserting rows using "values"

Proc sql;
Create table megh2 as selecct * from sasuser.admit (keep = id name sex age);
quit;

proc sql;
insert into megh2
values ('1234','amit','M',29) /*as id is char in nature, do not give any comma or semi colon */

values ('1245', 'Preeti','F',28);
quit;

Log display:

```
52    insert into megh2
53    values ('1234','amit','M',29)
54    values ('1245','Preeti','F',28);
NOTE: 2 rows were inserted into WORK.MEGH2.

55    quit;
NOTE: PROCEDURE SQL used (Total process time)
      real time           0.00 seconds
```

3. Inserting rows using query

Proc sql;
Create table megh3 like sasuser.admit;
Insert into megh3
Select * from sasuser.admit where sex='F';
quit;

Proc sql;
Create table megh3 as select * from sasuser.admit;
Insert into megh3
Select * from sasuser.admit where sex='F';
quit;

Adding rows with base sas:

Data a;
Set sasuser.admit(keep-id name sex);

```
If _n_=4 then do;
Output;
Id=”2178”;
Name=”Amit”;
Sex=”M”;
Output;
End;
Else output;
Run;
```

Updating rows with Keyword Update:

```
Proc sql;
Create table admit as select * from sasuser.admit;
quit;

proc sql;
update admit
set age=age*3,
height = height*10;
quit;
```

Log display:

ID	Name	Sex	Age	Date	Height	Wei
2458	Murray, W	M	32.4	1	72	
2462	Almers, C	F	40.8	3	66	
2501	Bonaventure, T	F	31	17	61	
2523	Johnson, R	F	43	31	63	
2539	LaMance, K	M	51	4	71	
2544	Jones, M	M	34.8	6	76	
2552	Reberson, P	F	32	9	67	
2555	King, E	M	35	13	70	
2563	Pitts, D	M	34	22	73	
2568	Eberhardt, S	F	49	27	64	
2571	Nunnelly, A	F	52.8	19	66	
2572	Oberon, M	F	28	17	62	
2574	Peterson, V	M	30	6	69	
2575	Quigley, M	F	48	8	69	

```
proc sql;                                          /* update with Case*/
update admit
set age=age*
case when actlevel=’HIGH’ then 1.2
          when actlevel=’MOD’ then 1.5
          when actlevel =’low’ then 2
 else 1                                  /* the else clause is necessary else rest
                                              of the values would be missing */
end;
quit;
```

Log display:

```
26   proc sql;
27   update lol
28   set age=age*
29   case when actlevel ='HIGH' then 1.2
30   when actlevel ='Mod' then 1.5
31   when actlevel='low' then 2
32   else 1
33   end;
NOTE: 21 rows were updated in WORK.LOL.
```

Monotonic function

This function is presented as a means to add sequence numbers to observations (rows) in a table.

Proc sql;
Select *,monotonic () as x from sasuser.admit;
quit;
Output:

D	Name	Sex	Age	Date	Height	Weight	Act Level	Fee	x
458	Murray, W	M	27	1	72	168	HIGH	85.20	1
462	Almers, C	F	34	3	66	152	HIGH	124.80	2
501	Bonaventure, T	F	31	17	61	123	LOW	149.75	3
523	Johnson, R	F	43	31	63	137	MOD	149.75	4
539	LaMance, K	M	51	4	71	158	LOW	124.80	5
544	Jones, M	M	29	6	76	193	HIGH	124.80	6
552	Reberson, P	F	32	9	67	151	MOD	149.75	7
555	King, E	M	35	13	70	173	MOD	149.75	8
563	Pitts, D	M	34	22	73	154	LOW	124.80	9
568	Eberhardt, S	F	49	27	64	172	LOW	124.80	10
571	Nunnelly, A	F	44	19	66	140	HIGH	149.75	11
572	Oberon, M	F	28	17	62	118	LOW	85.20	12
574	Peterson, V	M	30	6	69	147	MOD	149.75	13
575	Quigley, M	F	40	8	69	163	HIGH	124.80	14
578	Cameron, L	M	47	5	72	173	MOD	124.80	15
579	Underwood, K	M	60	22	71	191	LOW	149.75	16
584	Takahashi, Y	F	43	29	65	123	MOD	124.80	17
586	Derber, B	M	25	23	75	188	HIGH	85.20	18
588	Ivan, H	F	22	20	63	139	LOW	85.20	19
589	Wilcox, E	F	41	16	67	141	HIGH	149.75	20
595	Warren, C	M	54	7	71	183	MOD	149.75	21

proc sql;
select *,monotonic () as x from sasuser.admit having x=max(x);
quit;
Output

ID	Name	Sex	Age	Date	Height	Weight	Act Level	Fee	x
2595	Warren, C	M	54	7	71	183	MOD	149.75	21

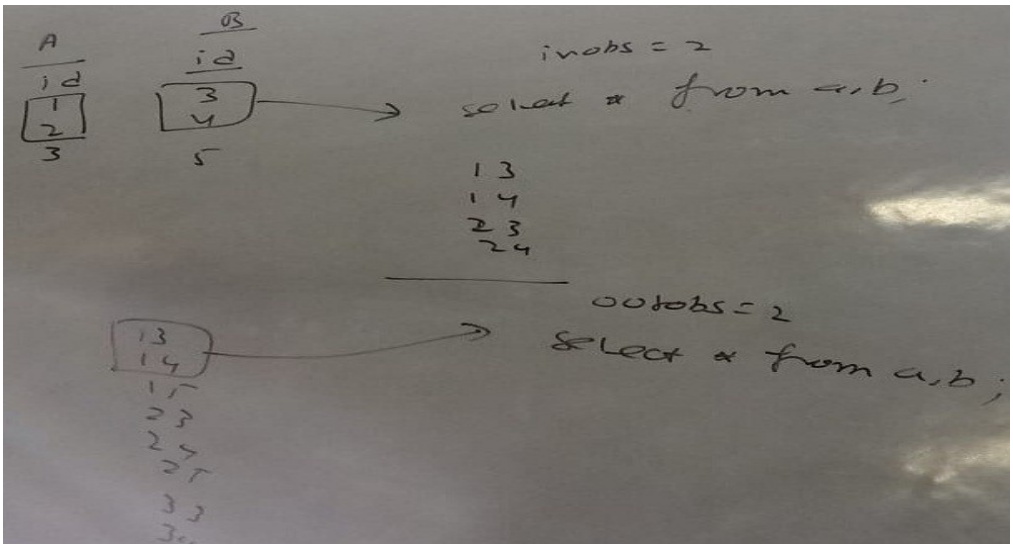
proc sql;
select *,monotonic () as x from sasuser.admit having x ge max(x)-1;
quit;
Output

ID	Name	Sex	Age	Date	Height	Weight	Act Level	Fee	x
2589	Wilcox, E	F	41	16	67	141	HIGH	149.75	20
2595	Warren, C	M	54	7	71	183	MOD	149.75	21

Inobs & Outobs

INOBS=n : This keyword restricts the no. of observations (rows) that are retrieved from any source.

OUTOBS=n : restricts the no. of observations (rows) in the output.




```
Proc sql inobs=5;
Select name,age from sasuser.admit;
quit;

Proc sql outobs=5;
Select name,age from sasuser.admit;
quit;
```

Output

The SAS System

Name	Age
Murray, W	27
Almers, C	34
Bonaventure, T	31
Johnson, R	43
LaMance, K	51

```
Proc sql inobs=5;
Select name, age from sasuser.admit where age gt 40;
quit;

Proc sql outobs=5;
Select name, age form sasuser.admit where age gt 40;
quit;
```

Output

The SAS System

Name	Age
Johnson, R	43
LaMance, K	51
Eberhardt, S	49
Nunnelly, A	44
Cameron, L	47

Query Debug

1. **Validate keyword:** writes message in the log that states if the query is valid else it writes the error messages in the log.

```
Proc sql;
Validate select name,age from sasuser.admit;
quit;
```

Log Display

```
157  validate select name,age from sasuser.admit;
NOTE: PROC SQL statement has valid syntax.
158  quit;
```

2. **Noexec keyword:** to check the syntax of the Sql query without actually executing it.

```
proc sql noexec;
select name,age from sasuser.admit;
quit;
```

Log Display

```
160  select name,age from sasuser.admit;
NOTE: Statement not executed due to NOEXEC option
161  quit;
NOTE: PROCEDURE SQL used (Total process time):
```

Some other examples

1.Selecting specific columns for the deletion, just use the delete keyword and the where condition

```
proc sql;
delete from admit where sex eq 'M';
quit;

proc sql;
delete from admit where sex eq 'M' and age gt 35;
quit;
```

2.Deleting all rows from a table

```
proc sql;
delete from admit;
quit;
```

3.Deleting a column from the table using alter and drop statements

```
proc sql;
```

```
create table admit as select * from sasuser.admit;
quit;

proc sql;
alter table admit drop age;
quit;

proc sql;
alter table admit drop name, age,height,weight;
quit;
```

4.To delete a table use the keyword drop;

```
Proc sql;
drop table admit;
quit;

Proc sql;
drop table admit, amit,kaka;
quit;
```

UNIX Commands:

1. **ls**: it is a command to list files in UNIX and UNIX-like OS. When invoked without any arguments, it lists the files in the current working directory.
2. **CD**: CD refers to change directory is a command-line OS shell command used to change the current working directory in OS
3. **PWD**: stands for print work directory. This command writes the full pathname of the current working directory
4. **CAL**: Cal command is a command line utitlity for displaying a calendar in the terminal.
5. **Whoami**: displays the username of the current user when invoked
6. **Mkdir**: this command is used to create directories. It can create multiple directories at once and also set permissions when creating the directory.
7. **More**: More is a command to view the contents of a text file.
8. **Nedit**: The Nirvana Editor, is a text editor .
9. **BASH**: this takes you to a bash interpreter which starts a bash process.
10. **Exit**: to exit out of the bash interpreter