CLASS - 7

Local dataset option

```
data a (drop = fee weight) b (keep = height sex) c
set sasuser.admit (drop = id);          ——— (1)
name1 = name;
drop name;          ———→ Global dataset option
run;                ——— (2)
```

Explanation

1. Drop = id will control the input.
2. drop name is a statement and this will impact dataset a, b and c.

Eg: In dataset 'a' → id, name, fee, weight will not come.

Both (1) and (2) are global dataset option but still statement (1) is better while optimizing code.

Prog-1

```
data a;
set sasuser.admit;
age-m = age*12;
where age-m gt 400;
run;
```

Prog-2

```
data a;
set sasuser.admit;
age-m = age*12;
where age*12 gt 400;
run;
```

Prog-1 - It will not run and show errow as new variable is not created yet which is used with where condition. i-e- (age-m).

Interview Question

If an interviewer asks for filter with 'where' only then in that case refer to prog 2 - Apply filter to existing variable only with the required condition.

Here the new variable 'age-m' is not used but the definition 'age*12' is used with 'where' and age greater than 400 will be displayed.

Program

```
data a;
set sasuser.admit;
x= age = 40;
run;
```

↓

This will create a boolean variable (x= age=40). Wherever the age will be 40, it will display '1' and in other cases '0'.

This is the substitute of 'if else if' but only creates a boolean variable. → Interview question

Other example

ID	Name	Sex	Age	Date	x
10	Neha	f	34	3	0
11	Priya	f	40	8	1
12	Gaurav	M	41	4	0
13	Radha	f	40	5	1
14	Shyam	M	42	7	0

Another examples

```
data a;
set sasuser.admit;
x=age gt 40;
run;
```

```
data a;
set sasuser.admit;
x= actlevel IN ("HIGH" "LOW");
run;
```

```
data a;
set sasuser.admit;
x=age gt 40;
y= age gt 50;
if x=1 and y=1;
run;
```

All the records greater than 40 & 50 will display the result as '1'.

can be written as - `if x and y;`

It will by default take the value '1'.

Boolean Assignment (By group)

```
proc sort data = sasuser.admit out=a;
by sex ;
run;
```

→ sorted by sex and output goes into 'a'.

```
ods html file = "C: —— path ——";
proc print data = a;
sum fee ;
by sex;
run;
ods html close;
```

→ ⑥

Note: whichever data set we want to print, it should be sorted first, only then we can use 'by'.

Here dataset 'a' is sorted by sex. So, in the output two reports will be generated. female report is generated separately with the fee total and male report is generated separately with fee total and a grand total at the end.

① Here if we use 'by age', then it will not run as dataset 'a' is not sorted by age, it is sorted by sex.

Another Example

```
proc sort data = sashuser.cargoinv out=a;
by route;
run;

ods html file = "_____";
proc print data = a;
sum revcargo;
by route;
where revcargo gt 5000;
run;
ods html close;
```

Explanation → Here different reports of route like route 1, route 2 --- route 6 will be generated with respective sum of revcargo and a grand total at the end.

Removing Duplicates

```
data a;
input roll no. name$ age;
cards;
1  A  10
2  B  20
2  C  25
3  K  25
4  K  30
5  K  35
5  K  37
5  K  37
;
run;

proc sort data = a  out= b  nodupkey ;
by name;
run;
```

Note: Nodupkey will remove duplicates of the key mention with 'by' statement.

Here, 'by' is used with name so, last four rows will be deleted as it contains the duplicate of names.

Another eg:

```
proc sort data = a  out = b noduplkey ;
by roll no name;
run;
```

Explanation : Here the combination of rollno. and name will be checked. As there is nothing common duplicate, so no row will be deleted.

Dupout → It will store the duplicates values removed.

```
proc sort data = a  out = b dupout = c noduplkey
by roll-no name age;
run;
```

Explanation : a = b+c

'c' will store the duplicates removed.