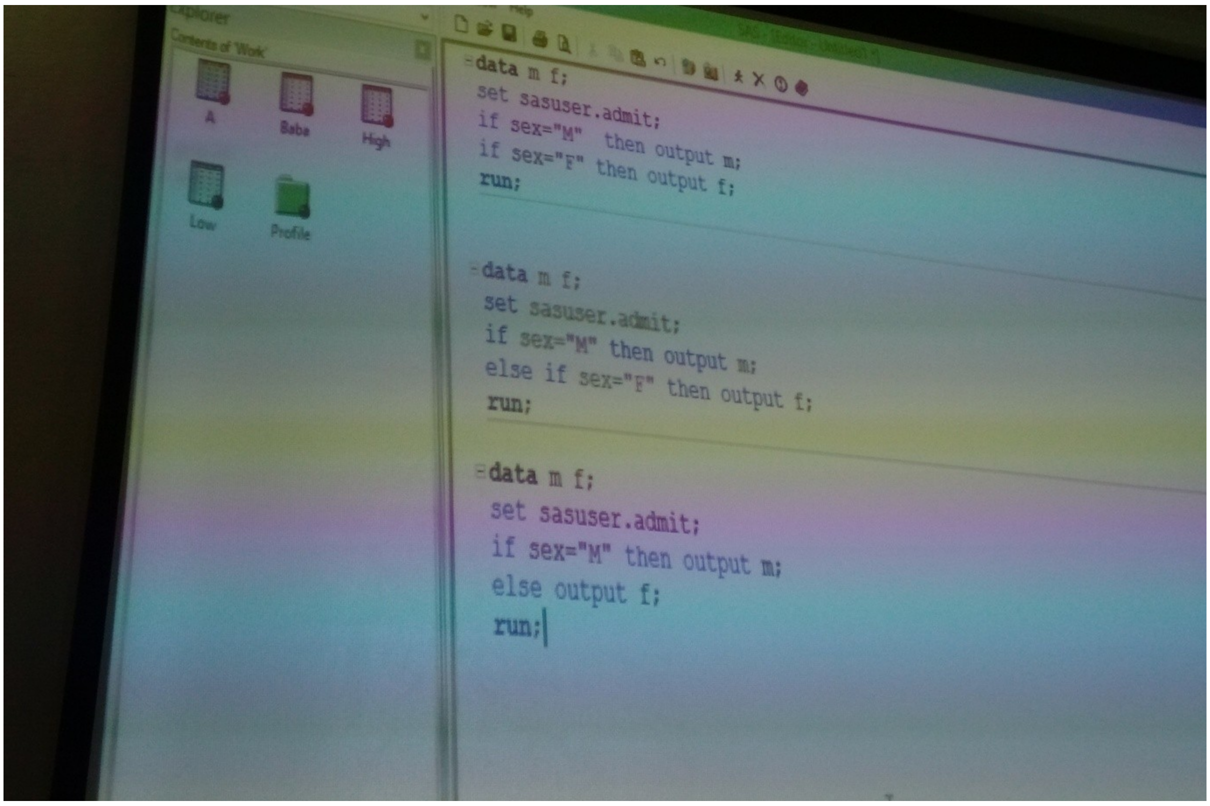
Dated: 10/03/2018

Statistical Analysis System : Class 6

..
..
..



```
data m f;
set sasuser.admit;
if sex="M"  then output m;
if sex="F" then output f;
run;

data m f;
set sasuser.admit;
if sex="M" then output m;
else if sex="F" then output f;
run;

data m f;
set sasuser.admit;
if sex="M" then output m;
else output f;
run;
```

If - Else-If Statement:

1. **Mutually exclusive if statement**:- multiple if statement are mutually exclusive and they would process data independently.

   Example:

   Data m f;

   Set sasuser.admit;

   If sex=”m” then output m;

   If sex=”f” then output f;

   Run;

   Explained: Two dataset “m” (male) and “ f ” (female) are created and where both IF statement are executed separately in two iterations.

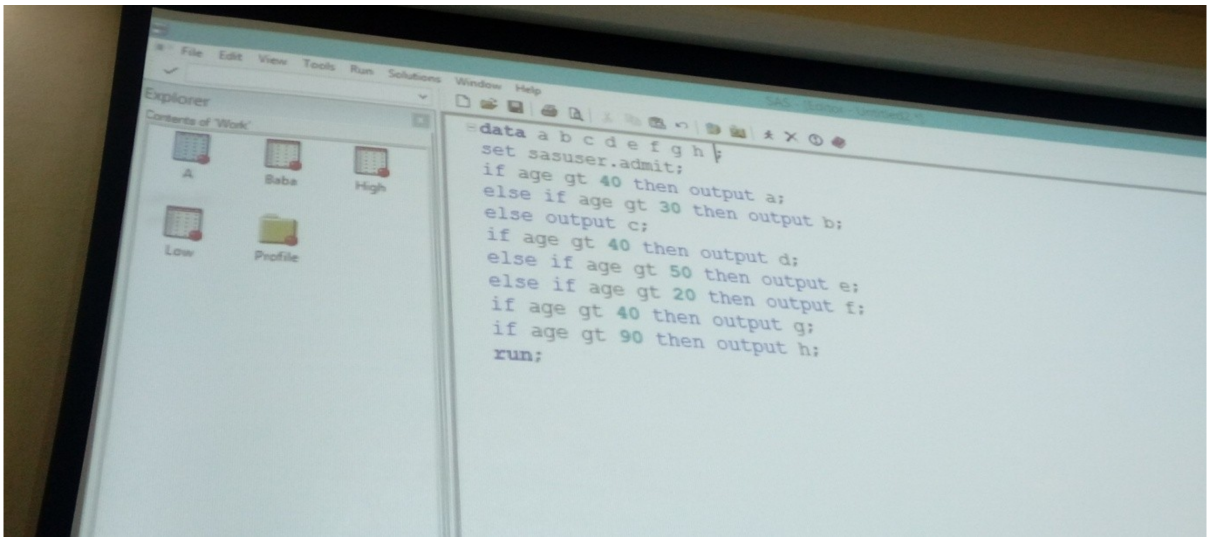
2. **<u>If Else If</u>** or nested if is more efficient as compared to multiple if statement because it can process entire data in a single iteration looking for different values of same variables.

   <u>Example</u>:
   Data m f;
   Set sasuser.admit;
   If sex = “m” then output m;
   Else if sex = “f” then output f;
   Run;

   <u>Explained</u>: two dataset “m” “f” created in single iteration where if statement and Else if statement are processed simultaneously.

3. **<u>If Else</u>** statement: It functions similar to If Else If statement with just one difference that only If statement works with logic being true, rest falls under category of Else.

   <u>Example:</u>
   Data m f;
   Set sasuser.admit;
   If sex = “m” then output m;
   Else output f;
   Run;
   <u>Explained</u>: logic to be true checked for only the If statement the rest values for the same variable will be under Else category.



```
data a b c d e f g h ;
set sasuser.admit;
if age gt 40 then output a;
else if age gt 30 then output b;
else output c;
if age gt 40 then output d;
else if age gt 50 then output e;
else if age gt 20 then output f;
if age gt 40 then output g;
if age gt 90 then output h;
run;
```

<u>Example for IF - ELSE – IF statements</u>:

Data a b c d e f g h ;

Set sasuser.admit;

 If age gt 40 then output a;

Else if age gt 30 then output b;

Else output c;

If age gt 40 then output d;

Else if age gt 50 the output e;

Else if age gt 20 the output f;

If age gt 40 then output g;

If age gt 90 then output h;

Run;

Explained: dataset ‘a’ has age with values greater to 40

Dataset ’b’ has age with values greater to 30 till 40.

‘c’ has age with values less than equal to 30.

‘d’ has values greater to 40.

‘e’ has values with age greater to 50 which is 0 values i.e null because all values greater to 40 are contained by ‘d’.

‘f’ has values greater to 20 till 40.

‘g’ has values greater to 40.

 ‘h’ has values greater to 90 which is again 0 values i.e null because value for age greater to 90 in the Admit dataset of sasuser library.

Reading data from FLAT files:-

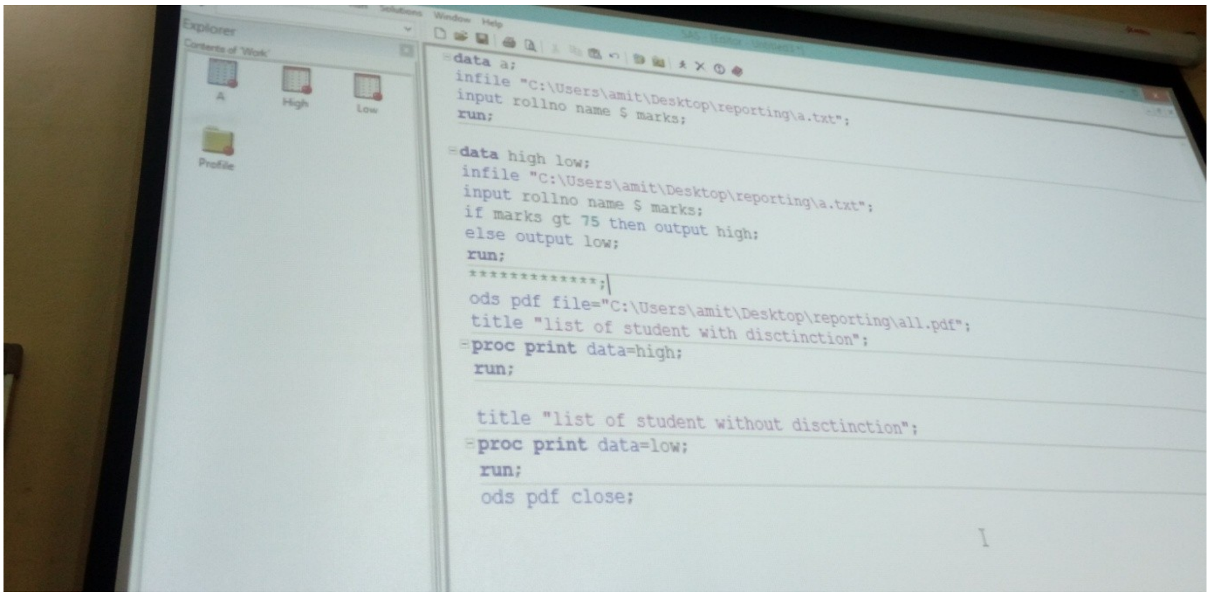
A FLAT file is a file that only stores data and not the attributes of data. INFILE is the keyword used to import FLAT files in SAS.

SYNTAX (INFILE): infile “path of the file“;

Two types of Flat files:

1. **Text file**: This is a normal notepad made file without any formatting.

Default delimiter of this file is space between the values.



```
data a;
infile "C:\Users\amit\Desktop\reporting\a.txt";
input rollno name $ marks;
run;

data high low;
infile "C:\Users\amit\Desktop\reporting\a.txt";
input rollno name $ marks;
if marks gt 75 then output high;
else output low;
run;
*************;
ods pdf file="C:\Users\amit\Desktop\reporting\all.pdf";
title "list of student with disctinction";
proc print data=high;
run;

title "list of student without disctinction";
proc print data=low;
run;
ods pdf close;
```

Example importing .TXT file (Test file):

Data a;

Infile “c:\users\amit\desktop\reporting\a.txt”;

Input rollno name $ marks;

Run;

Explained: dataset ‘a ‘ is created by using **“infile”** for importing contents from text file ‘a.txt’ and using keyword **“input”** for declaring variables rollno (numeric type) name $ (character type) marks (numeric type).

Other examples:

Data high low;

Infile “c:\users\amit\desktop\reporting\a.txt”;

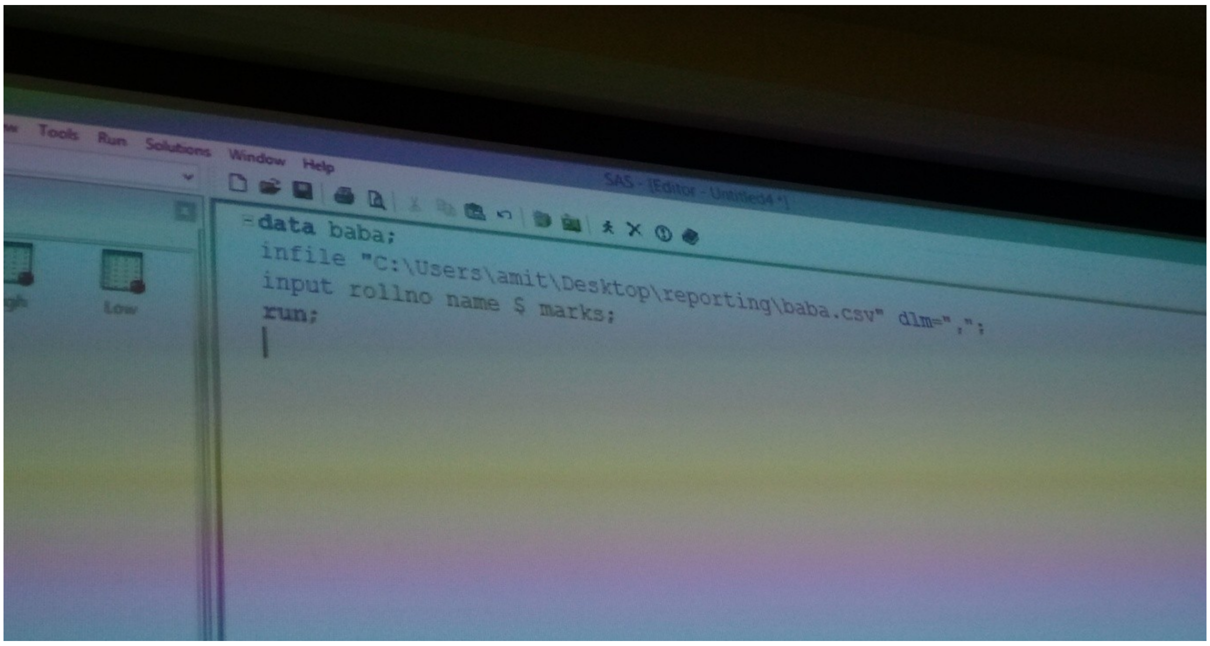
Input rollno name $ marks;

If marks gt 75 then output high;

Else output low;

Run;

Explained: Two dataset “high” and “low” are created where **infile** imports .txt file and **input** defines variables and if else statement filters out data respectively.



```
data baba;
infile "C:\Users\amit\Desktop\reporting\baba.csv" dlm=",";
input rollno name $ marks;
run;
```

2. **<u>.CSV File (CSV – Comma separated values)</u>**:
    - Is another kind of excel file but without any formatting saved on the .CSV file.
    - While working with .CSV file it is always necessary to use Delimiter.
    - Syntax for using delimiter : infile “path” dlm=”symbol”

Differentiating between .CSV file and .XLSX file:

- .CSV file has .csv extension in the name and excel file has .xslx extension and they can also be differentiated by looking at their icons.
- .CSV file has just 1 sheet unlike excel with 3 sheets when opened, also .CSV has a tab name similar to file name.
- If a .CSV file is opened as notepad, then values are separated by comma (,) that means delimiter is comma (,).

Example:

Data baba;

Infile”c:\users\amit\desktop\reporting\baba.csv” dlm=”,”;

Input rollno name $ marks;

Run;

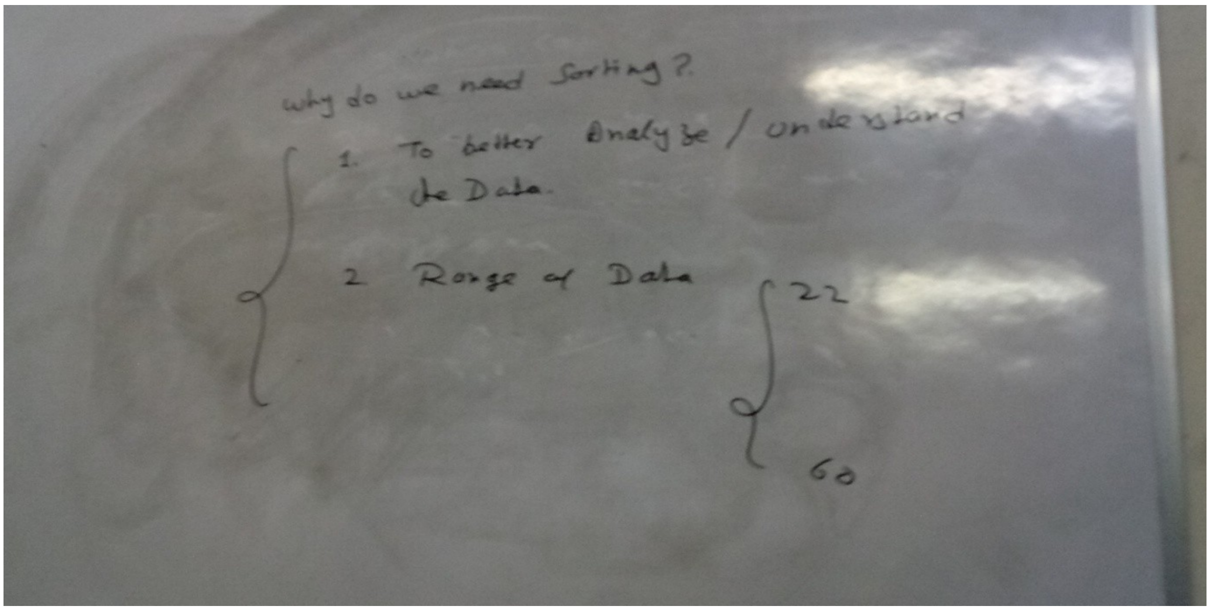
Explained: Dataset baba is created importing data from a .CSV file where delimiter is “,” i.e in the data values are separated by comma (,) with variables rollno (numeric type) name (character type) marks (numeric type).

PROC SORT:

This Proc function does sorting of available data. By default sorting is in ascending order.

Why do we need sorting?

1. To better analyse / understand data.
2. To find the range of data.



why do we need Sorting?

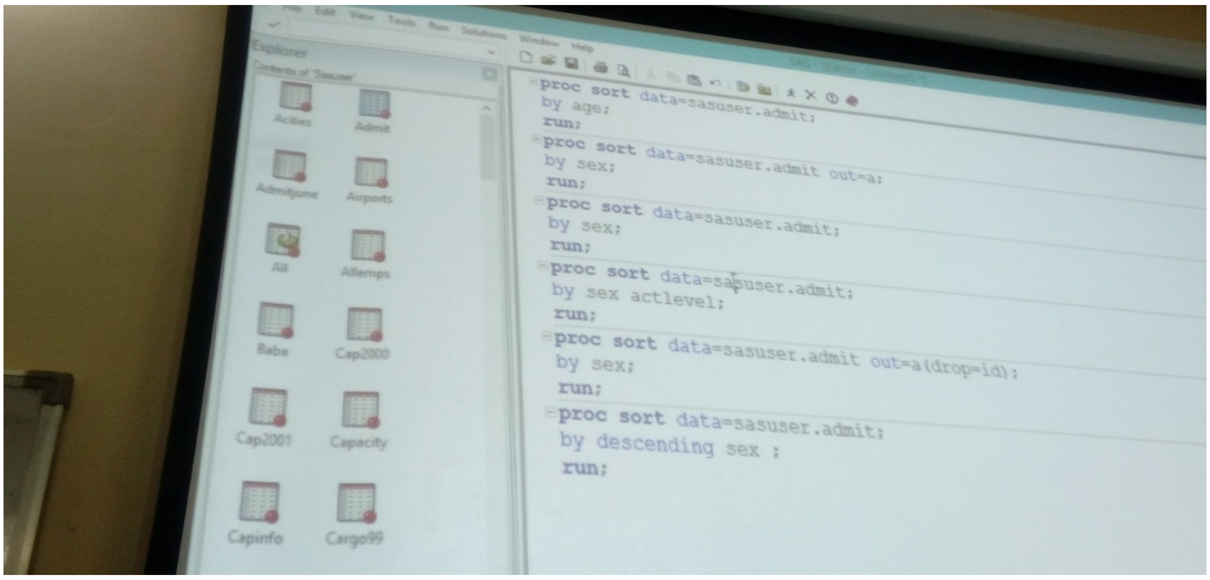
1. To better Analyze / understand the Data.

2. Range of Data

22

60

Some examples for PROC SORT:



```
proc sort data=sasuser.admit;
by age;
run;
proc sort data=sasuser.admit out=a;
by sex;
run;
proc sort data=sasuser.admit;
by sex;
run;
proc sort data=sasuser.admit;
by sex actlevel;
run;
proc sort data=sasuser.admit out=a(drop=id);
by sex;
run;
proc sort data=sasuser.admit;
by descending sex ;
run;
```

Example 1:
Proc sort data = sasuser.admit;
By age;
Run;

Explained: This code does sorting on original admit dataset using keyword "BY" implied on age variable, resulting in dataset admit being sorted with age from lowest to highest.

BY: This is another keyword used in PROC sort with variables to sort the data.

Example 2:
Proc sort data = sasuser.admit out = a;
By sex;
Run;

Explained: This code does sorting on admit dataset using keyword "BY" implied on sex variable, resulting in dataset "a" being sorted with sex Female first and Male second (in ascending alphabetical order) as all this is created on a new dataset "a" (implied by **OUT=a**)

OUT: To create a separate dataset after sorting from the original datasets .

Example 3:

Proc sort data = sasuser.admit;
By sex;
Run;

Explained: This code does sorting on original admit dataset using keyword "BY" implied on sex variable, resulting in dataset admit being sorted with sex Female first and Male second (in ascending alphabetical order).

Example 4:

```
Proc  sort data = sasuser.admit;
By sex actlevel;
Run;
```

Explained: This code does sorting on original admit dataset using keyword **“BY” implied 1st on sex variable**, resulting in dataset admit being sorted with sex Female first and Male second (in ascending alphabetical order) and 2nd on actlevel variable with values High first, Low second again in (in ascending alphabetical order).

Example 5:

```
Proc  sort data = sasuser.admit out = a (drop=id);
By sex;
Run;
```

Explained: This code does sorting on admit dataset using keyword “BY” implied on sex variable, resulting in dataset “a” being sorted with sex Female first and Male second (in ascending alphabetical order) but excluding variable “ID” compared to original dataset admit.

Example 6:

```
Proc  sort data = sasuser.admit;
By descending sex;
Run;
```

Explained: This code does sorting on orignal admit dataset using keyword “BY” implied on sex variable, resulting in admit dataset being sorted with sex Male first and Female second (in descending alphabetical order).