Day 2

\*\* SAS program consists of 2 parts  
  
1) Data step  
  
2) Proc step  
  
to handle the data  
  
- Program consists of SAS statments  
- programs are not case sensitive - can be lower case or upper case  
for example - below are same ;  
DATA TEST1;  
SET SASHELP.CLASS;  
RUN;  
  
data test2;  
set sashelp.class;  
run;  
  
\*\* each line is called a statement.;  
\*\*\*\*\* each statement should end with a semicolon  
\* more than one statement can be on one line or multiple lines \*\*;  
  
\*\* commenting code \*\*;  
\*\* you can do the comment in 2 ways \*\*;  
/\* 1) start with a star and ends with a semicolon  - refer to 2nd option below \*/  
\* 2) start with /\* and ends with a \*/ - refer to 1st option above ;  
  
proc print data=sashelp.class;  
run;  
  
\*\* referring to SAS datasets \*\*;  
\*\* one level vs two level \*\*;  
\*\* SAS datasets will be referred from permenant libraries or work libraries \*\*;  
\*\* permenant libraries like SAS provided libraries or user defined libraries using LIBNAME statement \*\*;  
\*\* if you don't refer from any library, then SAS will automatically assumes it is in WORK library \*\*;  
\*\* two level reference means - library.datasetname - the text before dot is a library, the text after dot is the dataset name \*\*;  
\*\* one level reference means - default referring to WORK library \*\*;  
  
\*\* accesssing data \*\*;  
\*\* we use LIBNAME to bring data to SAS session \*\*;  
libname sdtm "C:\Users\neela\Desktop\SAS training\SAS Stuff\SAS\_Data\SDTM";  
  
data ae;  
set [sdtm.ae](http://sdtm.ae/);  
run;  
  
data sdtm.test1;  
set test1;  
run;  
  
\*\* SAS datasets in permanant library stays there \*\*;  
\*\* SAS datasets in WORK library are temporary and will be available only until the SAS session is active,  
that mean once you close SAS session, the datasets in WORK will be erased \*\*;  
  
\*\* naming conventions \*\*;  
\*\* - SAS dataset names and Variable names  
- 32 characters or less  
- it should contain only letters, numbers or underscore, no special characters are allowed  
- it should start with an underscore or a characters (AtoZ)  
- it should not start with any numbers  
  
\*\*;  
  
\*\* single quotes vs double quotes \*\*;  
\*\* when you are referring to the path or a text you have to use either single quote or double quotes \*\*;

Day 3

data class;  
set sashelp.class;  
run;  
  
data classfit;  
set sashelp.classfit;  
run;  
  
\*\* in Data step commonly used things are -  
\*\* Set statement - this is used to copy data from one location to another location \*\*;  
\*\* merge statement - this is to combine datasets \*\*;  
  
\*\* set copies data vertically \*\*;  
\*\* merge copies data horizontally and them vertically \*\*;  
  
data class2;  
set sashelp.class sashelp.classfit;  
run;  
  
\*\* what do we do for a dataset -  
1) we may need to filter the dataset  
- Row level filtering - example: we have 100 observation in the input data, but need only 50 observations  
- Column level filtering - example: we have 10 variables, but need only 4 variables or 8 variables  
2) Sorting  
3) Add new variables  
4) Rename variables  
5) Modify the values in the observations  
- character modification  
- formatting  
- calculations  
\*\*;  
  
\*\* 1) filtering the data in row level \*\*;  
\*\* - we can use DATA step  
\*\* there are 2 options to filter the data in row level  
use IF or WHERE statements \*\*;  
\*\* when referring to values in character variables, we need to use quotes, single or double quotes \*\*;  
data class\_m;  
set sashelp.class;  
if sex = 'M' or age = 15;  
if sex in ('M' , 'F');  
run;  
  
\*\* when referring to values in numeric variable, we should not use quotes \*\*;  
data class\_15\_if;  
set sashelp.class;  
if age = 15;  
run;  
  
data class\_15\_where;  
set sashelp.class;  
where age = 15;  
run;  
  
\*\* IF - reads the data, writes the data, then filters the data \*\*;  
\*\* WHERE - reads the data, filters the data, then writes the data \*\*;  
  
\*\* operators are used in IF or WHERE \*\*;  
\*\* >, <, =, >=, <=, and, or, between, in \*\*;

Day 4

\*\* what do we do for a dataset -  
1) we may need to filter the dataset  
- Row level filtering - example: we have 100 observation in the input data, but need only 50 observations  
- Column level filtering - example: we have 10 variables, but need only 4 variables or 8 variables  
2) Sorting  
3) Add new variables  
4) Drop some variables  
5) Rename variables  
6) Modify the values in the observations  
- character modification  
- formatting  
- calculations  
\*\*;  
  
\*\* 1) filtering the data in row level \*\*;  
\*\* - we can use DATA step  
\*\* there are 2 options to filter the data in row level  
use IF or WHERE statements \*\*;  
\*\* when referring to values in character variables, we need to use quotes, single or double quotes \*\*;  
data class\_m;  
set sashelp.class;  
if sex = 'M' or age = 15;  
if sex in ('M' , 'F');  
run;  
  
\*\* when referring to values in numeric variable, we should not use quotes \*\*;  
data class\_15\_if;  
set sashelp.class;  
if age = 15;  
run;  
  
data class\_15\_where;  
set sashelp.class;  
where age = 15;  
run;  
  
\*\* IF - reads the data, writes the data, then filters the data \*\*;  
\*\* WHERE - reads the data, filters the data, then writes the data \*\*;  
  
\*\* operators are used in IF or WHERE \*\*;  
\*\* >, <, =, >=, <=, and, or, between, in \*\*;  
  
  
\*\* 2) filtering the data in column level \*\*;  
\*\* we use KEEP or DROP \*\*;  
\*\* use as STATEMENT or OPTION \*\*;  
  
data class;  
set sashelp.class;  
run;  
  
\*\* using as STATEMENT \*\*;  
data age1;  
set class;  
keep name age;  
run;  
  
data age2;  
set class;  
drop sex height weight;  
run;  
  
\*\* using as a DATA STEP OPTION \*\*;  
\*\* when used in the dataset used in SET statement, SAS first filters the data and then writes the data \*\*;  
data age3 ;  
set class(keep=name age);  
run;  
  
\*\* when used in the dataset used in DATA statement, SAS first writes the data and then filters the data \*\*;  
data age3 (keep=name age);  
set class;  
run;  
  
  
\*\* SORTING \*\*;  
\*\*\*\*\*\* Allways accompanied by a BY statement \*\*;  
PROC SORT data=class;  
by height;  
run;  
  
data test;  
set class;  
run;  
  
\*\* if you want to output a dataset after sorting \*\*;  
\*\* the input dataset remains same when you use OUT= option in PROC SORT statement \*\*;  
data class;  
set sashelp.class;  
run;  
  
\*\* default is always ascending \*\*;  
proc sort data=class out=height\_ascending;  
by height;  
run;  
  
\*\* if you want to sort by descending, specify before variable in BY statment \*\*;  
proc sort data=class out=height\_descending;  
by descending height;  
run;  
  
\*\* if you want to remove the duplicates - use NODUPKEY option \*\*;  
data class;  
set sashelp.class;  
run;  
  
proc sort data=sashelp.class out=age1 (keep=name age) nodupkey;  
by age;  
where sex = "F";  
run;  
  
\*\* IF can be used only in DATA step \*\*;  
\*\* WHERE can be used in both DATA and PROC steps \*\*;  
  
\*\* 1) first sas reads SASHELP.CLASS \*\*;  
\*\* 2) filters data where SEX = 'F' using WHERE statement \*\*;  
\*\* 3) sorts data by AGE using BY statement \*\*;  
\*\* 4) writes the data to AGE1 dataset using OUT= option \*;  
\*\* 5) deletes duplicates in the output dataset using NODUPKEY \*\*;  
\*\* 6) column level filtering in the output dataset using KEEP = option \*\*;