Statistical Analysis System: Class 17
Dated: 28 Apr, 2018

Global & Local Options in SAS

- If defined once, works for all the datasets in the sas editor until the session is closed
- If global option is overwritten, then updated global option will be functional for the rest of the code in the editor.
- If a datastep has local option, the same will be functional only and not the global option.
- By default global option has MAX range.
- Practical application is the SRP (sample run of production).

SAS Program	Explanation

Prog 1
options obs=2;
data a;
set sasuser.admit;
run;
	Explained For 1
	Global option defined is OBS = 2
	Output dataset "a" will have only 2 Observations.
	NOTE: There were 2 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.A has 2 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 1.56 seconds
	cpu time 1.40 seconds

Prog 2
options obs=5;
data b;
set sasuser.admit;
run;
	Explained For 2
	Global option overwritten, is OBS = 5
	Output dataset "b" will have only 5 Observations.
	NOTE: There were 5 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.B has 5 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 0.07 seconds
	cpu time 0.04 seconds

Prog 3
data c;
set
sasuser.admit(obs=10)
;
run;
	Explained For 3
	Global option overwritten, is set to OBS = 5 (from Prog 2).
	But Local option sets, OBS=10 and nullifies Global option, OBS=5, but only for this particular datastep.
	Output dataset "c" will have 10 Observations.
	NOTE: There were 10 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.C has 10 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 0.04 seconds
	cpu time 0.04 seconds

Prog 4
data k;
set sasuser.admit;
run;
	Explained For 4
	Global option, OBS=5 (continues from Prog 2)
	Output "K" has 5 observations.
	NOTE: There were 5 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.K has 5 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 0.04 seconds
	cpu time 0.04 seconds

Prog 5
`options obs=max;`
`data d;`
`set sasuser.admit;`
`run;`
	Explained For 5
	Global option, OBS=max (Obs=max, is the default range for global option)
	Output "d" has 21 observations.
	NOTE: There were 21 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.D has 21 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 0.03 seconds
	cpu time 0.03 seconds

Prog 6
`data e;`
`set sasuser.admit;`
`run;`
	Explained For 6
	Global option, OBS=max (Obs=max, continued from Prog 5)
	Output "e" has 21 observations.
	NOTE: There were 21 observations read from the data set SASUSER.ADMIT.
	NOTE: The data set WORK.E has 21 observations and 9 variables.
	NOTE: DATA statement used (Total process time):
	real time 0.03 seconds
	cpu time 0.03 seconds

PROC SQL:

- SQL stands for Structured Query Language.
- Query is a code written with SQL
- Proc sql goes with following 6 keywords:
    1. Select (select *, denotes selection of every variable from the specified table)
    2. From
    3. Where
    4. Group By
    5. Having
    6. Order By

/* Prog 7: Simple sql query for creating a table */

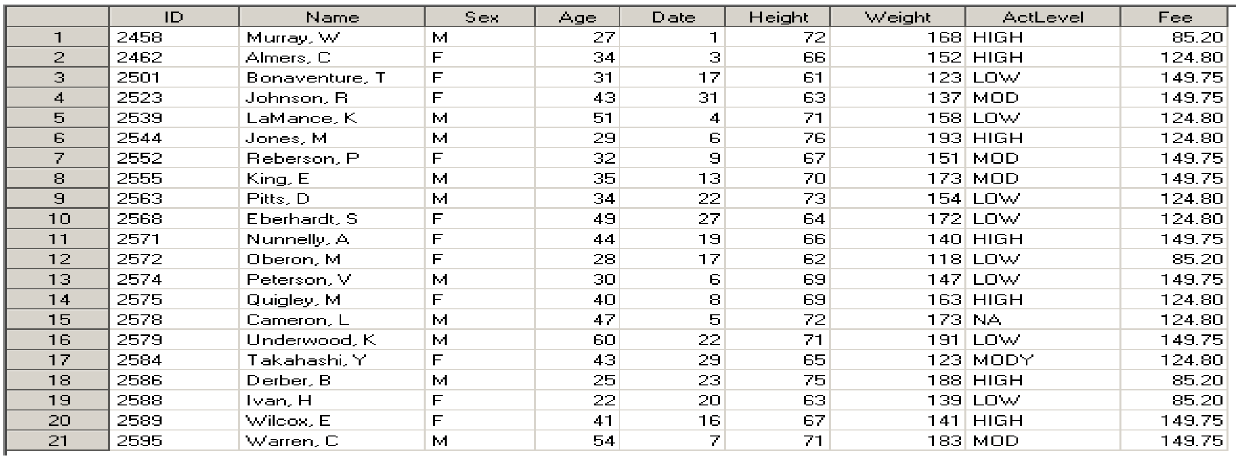
Sql Query	Base SAS equivalent (datastep / Procstep)

`proc sql;`
`create table a as select * from sasuser.admit;`
`quit;`
	`data a;`
	`set sasuser.admit;`
	`run;`

Explained: This Sql Query will create a table "a" with all the variables and all observations from S.A.

```
NOTE: Table WORK.A1 created, with 21 rows and 9 columns.
NOTE: PROCEDURE SQL used (Total process time):
      real time          1.23 seconds
      cpu time           1.14 seconds
```

OUTPUT:



	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee
1	2458	Murray, W	M	27	1	72	168	HIGH	85.20
2	2462	Almers, C	F	34	3	66	152	HIGH	124.80
3	2501	Bonaventure, T	F	31	17	61	123	LOW	149.75
4	2523	Johnson, R	F	43	31	63	137	MOD	149.75
5	2539	LaMance, K	M	51	4	71	158	LOW	124.80
6	2544	Jones, M	M	29	6	76	193	HIGH	124.80
7	2552	Reberson, P	F	32	9	67	151	MOD	149.75
8	2555	King, E	M	35	13	70	173	MOD	149.75
9	2563	Pitts, D	M	34	22	73	154	LOW	124.80
10	2568	Eberhardt, S	F	49	27	64	172	LOW	124.80
11	2571	Nunnelly, A	F	44	19	66	140	HIGH	149.75
12	2572	Oberon, M	F	28	17	62	118	LOW	85.20
13	2574	Peterson, V	M	30	6	69	147	LOW	149.75
14	2575	Quigley, M	F	40	8	69	163	HIGH	124.80
15	2578	Cameron, L	M	47	5	72	173	NA	124.80
16	2579	Underwood, K	M	60	22	71	191	LOW	149.75
17	2584	Takahashi, Y	F	43	29	65	123	MODY	124.80
18	2586	Derber, B	M	25	23	75	188	HIGH	85.20
19	2588	Ivan, H	F	22	20	63	139	LOW	85.20
20	2589	Wilcox, E	F	41	16	67	141	HIGH	149.75
21	2595	Warren, C	M	54	7	71	183	MOD	149.75

/* Prog 8: Subsetting through sql */

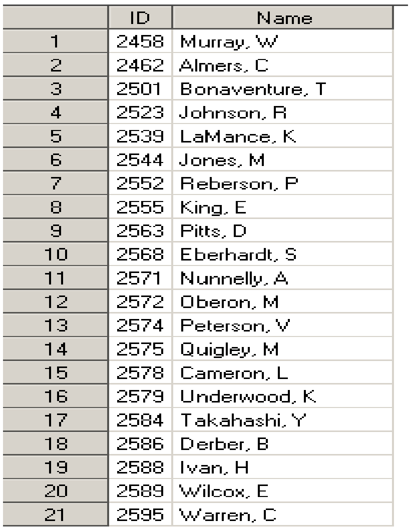
Sql Query	**Base SAS equivalent (datastep / Procstep)**

`proc sql;`
`create table b as select id,name`
`from sasuser.admit;`
`quit;`
	`data b (keep=id name);`
	`set sasuser.admit;`
	`run;`

Explained: This SQL query will create a table "b" but selecting only 2 variables (id, name) given after SELECT keyword from S.A.

```
NOTE: Table WORK.B created, with 21 rows and 2 columns.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time
```

OUTPUT:



	ID	Name
1	2458	Murray, W
2	2462	Almers, C
3	2501	Bonaventure, T
4	2523	Johnson, R
5	2539	LaMance, K
6	2544	Jones, M
7	2552	Reberson, P
8	2555	King, E
9	2563	Pitts, D
10	2568	Eberhardt, S
11	2571	Nunnelly, A
12	2572	Oberon, M
13	2574	Peterson, V
14	2575	Quigley, M
15	2578	Cameron, L
16	2579	Underwood, K
17	2584	Takahashi, Y
18	2586	Derber, B
19	2588	Ivan, H
20	2589	Wilcox, E
21	2595	Warren, C

/* Prog 9: Print through sql */

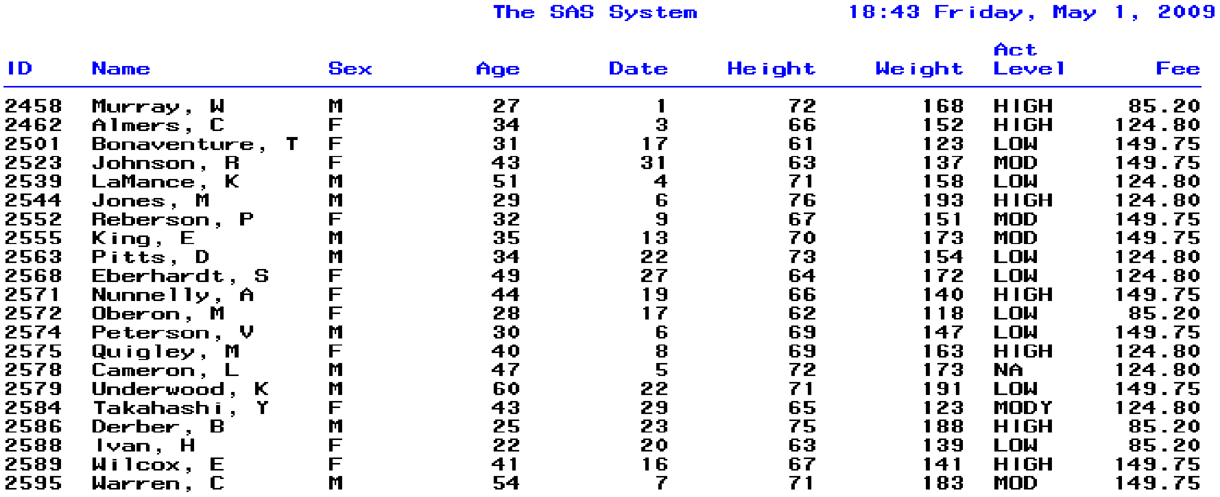
Sql Query	Base SAS equivalent (datastep / Procstep)

proc sql;
select * from sasuser.admit;
quit;
	proc print data=sasuser.admit;
	run;

Explained: This SQL query will print a table of the contents of S.A

NOTE: PROCEDURE SQL used (Total process time):
real time 0.37 seconds
cpu time 0.35 seconds

OUTPUT:



```
                                     The SAS System           18:43 Friday, May 1, 2009

                                                                          Act
 ID     Name               Sex        Age       Date    Height    Weight  Level      Fee

 2458   Murray, W          M          27          1       72       168   HIGH      85.20
 2462   Almers, C          F          34          3       66       152   HIGH     124.80
 2501   Bonaventure, T     F          31         17       61       123   LOW      149.75
 2523   Johnson, R         F          43         31       63       137   MOD      149.75
 2539   LaMance, K         M          51          4       71       158   LOW      124.80
 2544   Jones, M           M          29          6       76       193   HIGH     124.80
 2552   Reberson, P        F          32          9       67       151   MOD      149.75
 2555   King, E            M          35         13       70       173   MOD      149.75
 2563   Pitts, D           M          34         22       73       154   LOW      124.80
 2568   Eberhardt, S       F          49         27       64       172   LOW      124.80
 2571   Nunnelly, A        F          44         19       66       140   HIGH     149.75
 2572   Oberon, M          F          28         17       62       118   LOW       85.20
 2574   Peterson, V        M          30          6       69       147   LOW      149.75
 2575   Quigley, M         F          40          8       69       163   HIGH     124.80
 2578   Cameron, L         M          47          5       72       173   NA       124.80
 2579   Underwood, K       M          60         22       71       191   LOW      149.75
 2584   Takahashi, Y       F          43         29       65       123   MODY     124.80
 2586   Derber, B          M          25         23       75       188   HIGH      85.20
 2588   Ivan, H            F          22         20       63       139   LOW       85.20
 2589   Wilcox, E          F          41         16       67       141   HIGH     149.75
 2595   Warren, C          M          54          7       71       183   MOD      149.75
```

/* Prog 10: Selective print through sql */

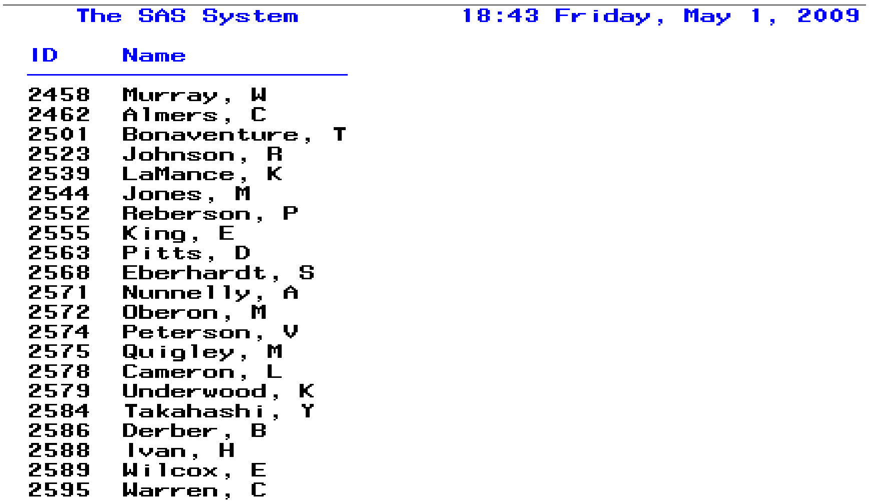
Sql Query	Base SAS equivalent (datastep / Procstep)

proc sql;
select id,name from sasuser.admit;
quit;
	proc print data=sasuser.admit;
	var id name;
	run;

Explained: This SQL query will print a table from S.A but only with variables id, name as stated after select keyword.

NOTE: PROCEDURE SQL used (Total process time):
real time 0.00 seconds
cpu time 0.00 seconds

OUPTUT:



```
     The SAS System            18:43 Friday, May 1, 2009

  ID     Name

 2458   Murray, W
 2462   Almers, C
 2501   Bonaventure, T
 2523   Johnson, R
 2539   LaMance, K
 2544   Jones, M
 2552   Reberson, P
 2555   King, E
 2563   Pitts, D
 2568   Eberhardt, S
 2571   Nunnelly, A
 2572   Oberon, M
 2574   Peterson, V
 2575   Quigley, M
 2578   Cameron, L
 2579   Underwood, K
 2584   Takahashi, Y
 2586   Derber, B
 2588   Ivan, H
 2589   Wilcox, E
 2595   Warren, C
```

/* **Prog 11: Selective print through sql** */

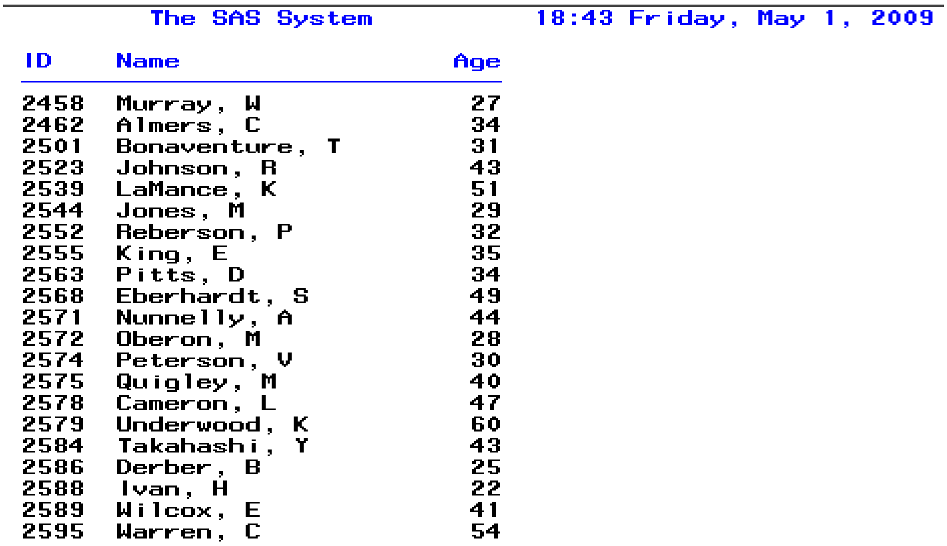
Sql Query	**Base SAS equivalent (datastep / Procstep)**

`proc sql;`
`select id,name,age from`
`sasuser.admit;`
`quit;`
	`proc print data=sasuser.admit;`
	`var id name age;`
	`run;`

Explained: Just like Prog 10, this will give variables: id, name,age as stated after select keyword.

NOTE: PROCEDURE SQL used (Total process time):
    real time   0.00 seconds
    cpu time   0.00 seconds

OUTPUT:



```
              The SAS System            18:43 Friday, May 1, 2009

 ID     Name                   Age

 2458   Murray, W               27
 2462   Almers, C               34
 2501   Bonaventure, T          31
 2523   Johnson, R              43
 2539   LaMance, K              51
 2544   Jones, M                29
 2552   Reberson, P             32
 2555   King, E                 35
 2563   Pitts, D                34
 2568   Eberhardt, S            49
 2571   Nunnelly, A             44
 2572   Oberon, M               28
 2574   Peterson, V             30
 2575   Quigley, M              40
 2578   Cameron, L              47
 2579   Underwood, K            60
 2584   Takahashi, Y            43
 2586   Derber, B               25
 2588   Ivan, H                 22
 2589   Wilcox, E               41
 2595   Warren, C               54
```

/* **Prog 12**: **Drop in SQL*/**

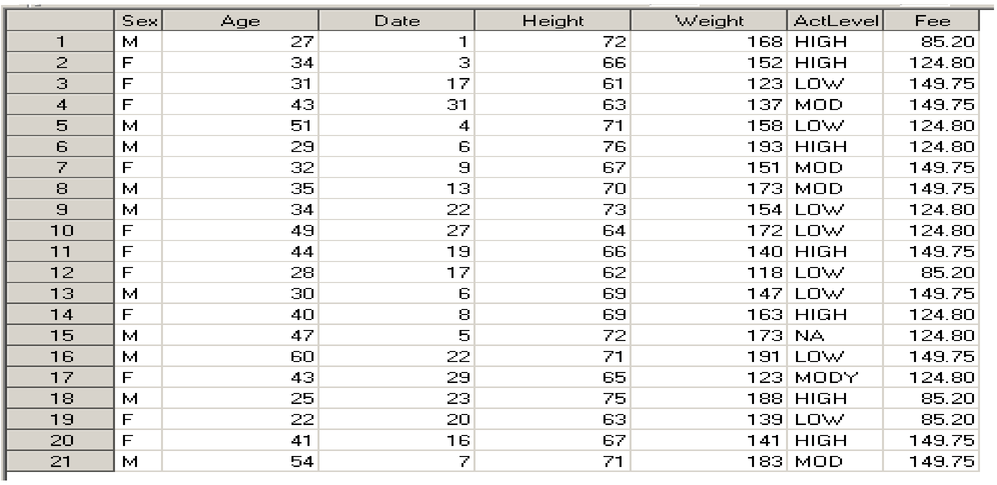
Sql Query	**Base SAS equivalent (datastep / Procstep)**

`proc sql;`
`create table a as select * from sasuser.admit(drop=id name);`
`quit;`
	`data a(drop=id name);`
	`set sasuser.admit;`
	`run;`

Explained: Drop, can be used as dataset option in Proc SQL to filter unwanted variables from any table, here id, name are dropped.

```
NOTE: Table WORK.A created, with 21 rows and 7 columns.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

OUTPUT:



	Sex	Age	Date	Height	Weight	ActLevel	Fee
1	M	27	1	72	168	HIGH	85.20
2	F	34	3	66	152	HIGH	124.80
3	F	31	17	61	123	LOW	149.75
4	F	43	31	63	137	MOD	149.75
5	M	51	4	71	158	LOW	124.80
6	M	29	6	76	193	HIGH	124.80
7	F	32	9	67	151	MOD	149.75
8	M	35	13	70	173	MOD	149.75
9	M	34	22	73	154	LOW	124.80
10	F	49	27	64	172	LOW	124.80
11	F	44	19	66	140	HIGH	149.75
12	F	28	17	62	118	LOW	85.20
13	M	30	6	69	147	LOW	149.75
14	F	40	8	69	163	HIGH	124.80
15	M	47	5	72	173	NA	124.80
16	M	60	22	71	191	LOW	149.75
17	F	43	29	65	123	MODY	124.80
18	M	25	23	75	188	HIGH	85.20
19	F	22	20	63	139	LOW	85.20
20	F	41	16	67	141	HIGH	149.75
21	M	54	7	71	183	MOD	149.75

/* **Prog 13**: **Multiple Table creation in SQL */**

Sql Query	**Base SAS equivalent (datastep / Procstep)**

`proc sql;`
`create table a as select * from sasuser.admit;`
`create table b as select * from sasuser.admit;`
`create table c as select * from sasuser.admit;`
`quit;`
	`data a b c;`
	`set sasuser.admit;`
	`run;`
	
	Note: for Multiple table creation Base SAS is more efficient as can be seen by the no of lines in the codes to avoid writing repetitive queries in SQL .

Explained: All the 3 datasets (a,b,c) will have all varaibles and observations from S.A.
OUTPUT: ouptut will be same as Prog 7 above for all 3 datasets (a,b,c).

/* Prog 14: Multiple Table creation in SQL*/

Sql Query	Base SAS equivalent (datastep / Procstep)

`proc sql;`
`create table a as select * from`
`sasuser.admit;`
`create table b as select id,name`
`from sasuser.admit;`
`create table c as select * from`
`sasuser.admit(drop=age);`
`quit;`
	`data a b(keep=id name) c(drop=age);`
	`set sasuser.admit;`
	`run;`

Explained: Dataset a, will have 21 obs and 9 variables from S.A
Dataset b, will have id, name as the only 2 variables and 21 observations from S.A.
Dataset c, will have 8 variables (excluding "age") and 21 observations from S.A.

OUPTUT:
Ouput dataset a, is same as output for Prog 7 above.
Ouput dataset b, is same as output for Prog 10 above.
Ouput dataset c, is same as Prog 7 above, except the "age" variable since age is dropped in dataset "c".

Creating new variable in SQL query is done as:
If a logical equation to create a new variable is suppose,
age_m = age*12;
Here, L.H.S = Age_m
R.H.S = age*12

SQL query will use R.H.S first and then L.H.S, considering that this new variable does not exist already in the parent dataset, therfore the SQL query becomes:

Create table (table_name) as select (specific variable name or *, can use as per the need),(R.H.S **o**f the logical equation) as (L.H.S / new variable_name) from (parent dataset name).

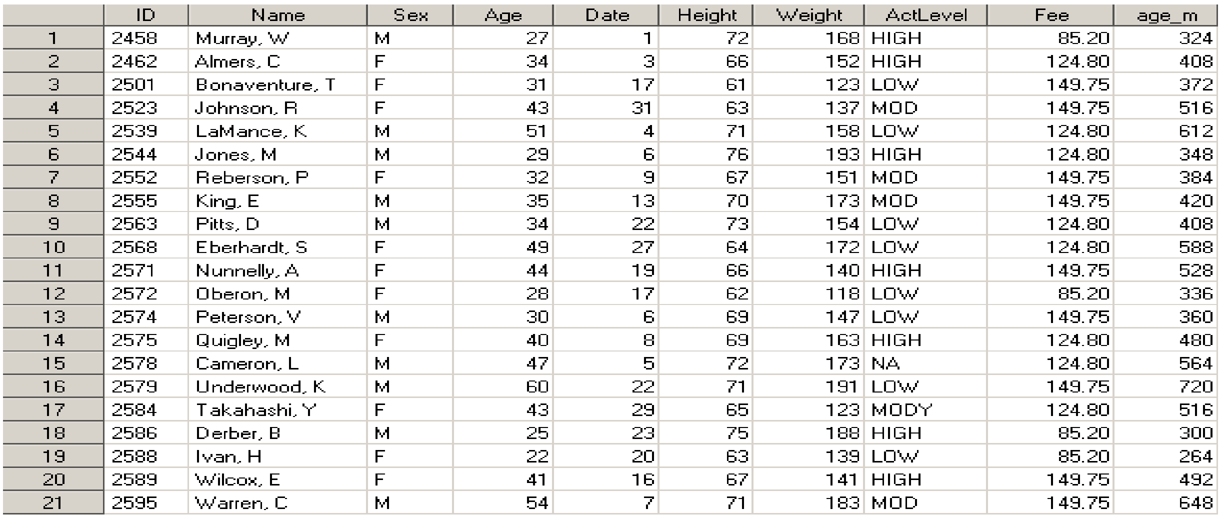
/* Prog 15: Creating new variable in SQL*/

Sql Query	Base SAS equivalent (datastep / Procstep)

`proc sql;`
`create table a as select *,age*12`
`as age_m from sasuser.admit;`
`quit;`
	`data a;`
	`set sasuser.admit;`
	`age_m=age*12;`
	`run;`

Explained: this query, creates a table with all the contents from S.A, adding a new variable, age_m to it.

OUTPUT:



	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee	age_m
1	2458	Murray, W	M	27	1	72	168	HIGH	85.20	324
2	2462	Almers, C	F	34	3	66	152	HIGH	124.80	408
3	2501	Bonaventure, T	F	31	17	61	123	LOW	149.75	372
4	2523	Johnson, R	F	43	31	63	137	MOD	149.75	516
5	2539	LaMance, K	M	51	4	71	158	LOW	124.80	612
6	2544	Jones, M	M	29	6	76	193	HIGH	124.80	348
7	2552	Reberson, P	F	32	9	67	151	MOD	149.75	384
8	2555	King, E	M	35	13	70	173	MOD	149.75	420
9	2563	Pitts, D	M	34	22	73	154	LOW	124.80	408
10	2568	Eberhardt, S	F	49	27	64	172	LOW	124.80	588
11	2571	Nunnelly, A	F	44	19	66	140	HIGH	149.75	528
12	2572	Oberon, M	F	28	17	62	118	LOW	85.20	336
13	2574	Peterson, V	M	30	6	69	147	LOW	149.75	360
14	2575	Quigley, M	F	40	8	69	163	HIGH	124.80	480
15	2578	Cameron, L	M	47	5	72	173	NA	124.80	564
16	2579	Underwood, K	M	60	22	71	191	LOW	149.75	720
17	2584	Takahashi, Y	F	43	29	65	123	MODY	124.80	516
18	2586	Derber, B	M	25	23	75	188	HIGH	85.20	300
19	2588	Ivan, H	F	22	20	63	139	LOW	85.20	264
20	2589	Wilcox, E	F	41	16	67	141	HIGH	149.75	492
21	2595	Warren, C	M	54	7	71	183	MOD	149.75	648

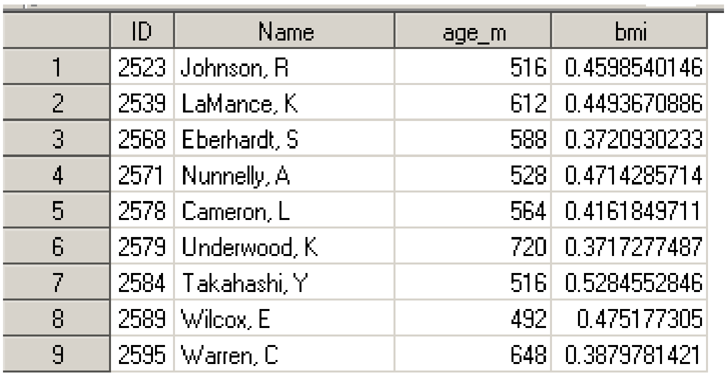
/* Prog 16: Creating new variable in SQL*/

Sql Query	Base SAS equivalent (datastep / Procstep)

`proc sql;`
`create table a as select`
`id,name,age*12 as age_m,height/weight`
`as bmi from sasuser.admit`
`where age gt 40;`
`quit;`
	`data a(keep=id name age_m bmi);`
	`set sasuser.admit;`
	`age_m=age*12;`
	`bmi=height/weight;`
	`where age gt 40;`
	`run;`

Explained: This query, creates a table with variables id, name, age_m, bmi from S.A while also checking for the condition of age > 40.

OUTPUT:



	ID	Name	age_m	bmi
1	2523	Johnson, R	516	0.4598540146
2	2539	LaMance, K	612	0.4493670886
3	2568	Eberhardt, S	588	0.3720930233
4	2571	Nunnelly, A	528	0.4714285714
5	2578	Cameron, L	564	0.4161849711
6	2579	Underwood, K	720	0.3717277487
7	2584	Takahashi, Y	516	0.5284552846
8	2589	Wilcox, E	492	0.475177305
9	2595	Warren, C	648	0.3879781421

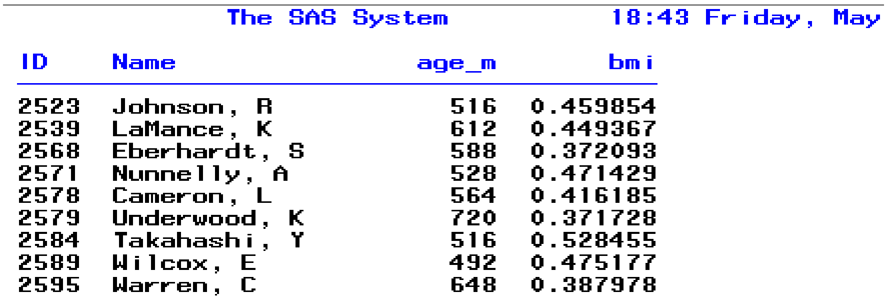
/* Prog 17: Sql equivalent for below 3 codes in Base Sas */

Sql Query	Base SAS equivalent (datastep / Procstep)

proc sql;
 	
create table a11 as select id,name,age*12 as age_m,height/weight as bmi from sasuser.admit
where age gt 40;
/* this statement Modifies dataset*/
 	
create table b as select * from a;
/* this statement creates new dataset*/
 	
select * from a;
/* this statement prints the dataset*/
 	
quit;
	data a(keep=id name age_m bmi);
	set sasuser.admit;
	age_m=age*12;
	bmi=height/weight;
	where age gt 40;
	run;
	....
	
	data b;
	set a;
	run;
	....
	
	proc print data=a;
	run;

Explained: This query explains how sql query simplifies, modifying a dataset, creating new dataset from the existing and also printing the dataset.

OUTPUT:



```
                The SAS System            18:43 Friday, May

 ID    Name                  age_m        bmi
 2523  Johnson, R              516   0.459854
 2539  LaMance, K              612   0.449367
 2568  Eberhardt, S            588   0.372093
 2571  Nunnelly, A             528   0.471429
 2578  Cameron, L              564   0.416185
 2579  Underwood, K            720   0.371728
 2584  Takahashi, Y            516   0.528455
 2589  Wilcox, E               492   0.475177
 2595  Warren, C               648   0.387978
```

/* **Prog 18**: **for the below Base SAS code while applying conditional formatting on a newly created variable, methods to write sql queries*/**

Note: Where does not work with a newly created variable within the datastep therefore If is used for any conditional formatting.

Base SAS Code:

```
data a;
set sasuser.admit;
age_m=age*12;
if age_m gt 400;
run;
```

SQL Queries:

Method 1: Applying conditional formatting (where) with the R.H.S of the logical equation.

```
proc sql;
create table a1 as select *,age*12 as age_m from sasuser.admit where
age*12 gt 400;
quit;
```

Method 2: Creating the new variable in 1st query in a table and then applying conditional formatting in the 2nd query..

```
proc sql;
create table a as select *,age*12 as age_m from sasuser.admit;
create table b as select * from a where age_m gt 400;
quit;
```

Method 3: Using **Calculated** Keyword with (where) for formatting.

```
proc sql;
create table a as select *,age*12 as age_m from sasuser.admit
where calculated age_m gt 400;
quit;
```

Method 4: **Inline view:** where view is a table but not physically present. This method first creates a table with the new variable in it (but this table has no physical presence) and then the where condition is applied on the new table created from the view table

```
proc sql;
create table a as select * from
(
select *,age*12 as age_m from sasuser.admit
)where age_m gt 400;
quit;
```