


### Learning outcomes:

1. Create and subset variables
2. Create observations in DATA step
3. Subset observations
4. Output multiple SAS data sets
5. Select variables and observations

### SAS components learnt:

1. SELECT group
  2. Subsetting IF statement
  3. Nested IF-THEN/ELSE statements
  4. KEEP and DROP options in SET and DATA statements
  5. WHERE statement
  6. FIRSTOBS= and OBS= options
- 

## 5.1 Creating and subsetting variables

 In Chapter 4, a variable can be created in a DATA step using assignment statement.



**Ex. 5.1:** The following table gives the values of Var1 and Var2 in Program Data Vector (PDV). Answer the questions.


Var1	Var2
.	20

In the SAS program:

```
...
Num1= (Var1+Var2)/4;
Num2=Sum(Var1,Var2)/4;
Num3=3+15/3;
...
```

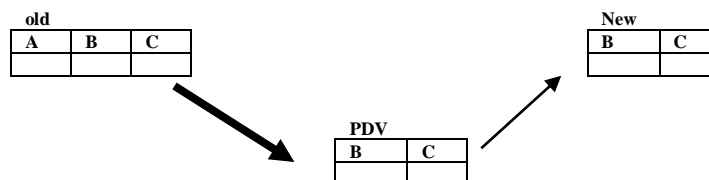
What is the value of

- (a) Num1?
- (b) Num2?
- (c) Num3?

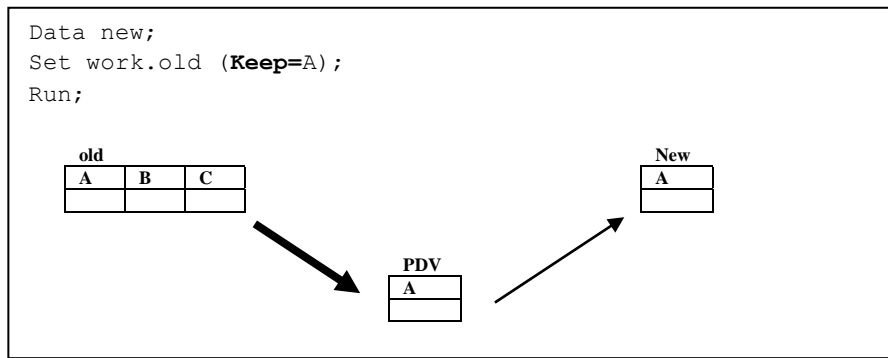
 Use KEEP= / DROP= options in SET statement to control the variable to be input from a data set


- Using DROP= options on an input data set:

```
Data new;
Set work.old (Drop=A);
Run;
```

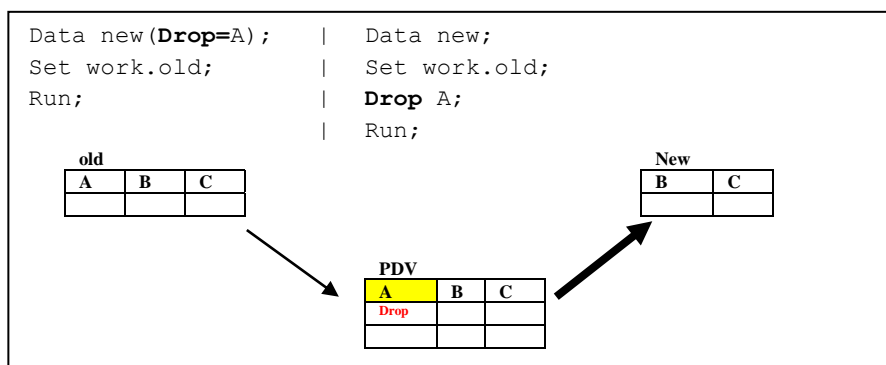


- Using KEEP= options on an input data set:

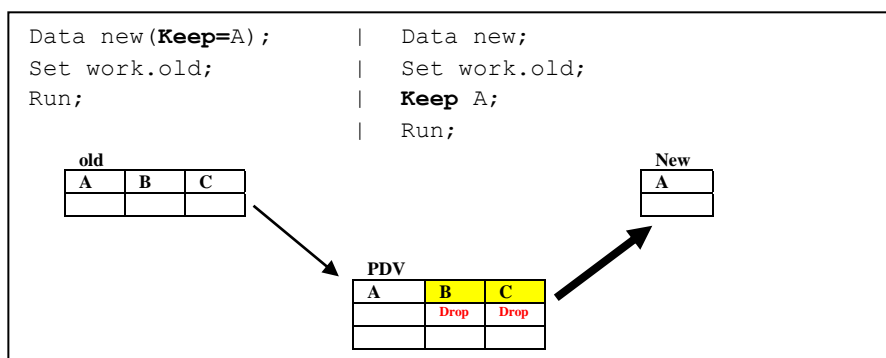



-  Use KEEP/DROP statements and KEEP/DROP options in DATA statement to control the variable to be output to a data set. The following two examples give the almost the same processing on PDV. Note that DROP statement is processed before the RUN statement (implicit return) while DROP= option is processed after the implicit return.

- Using DROP= option on an output data set / use DROP statement:

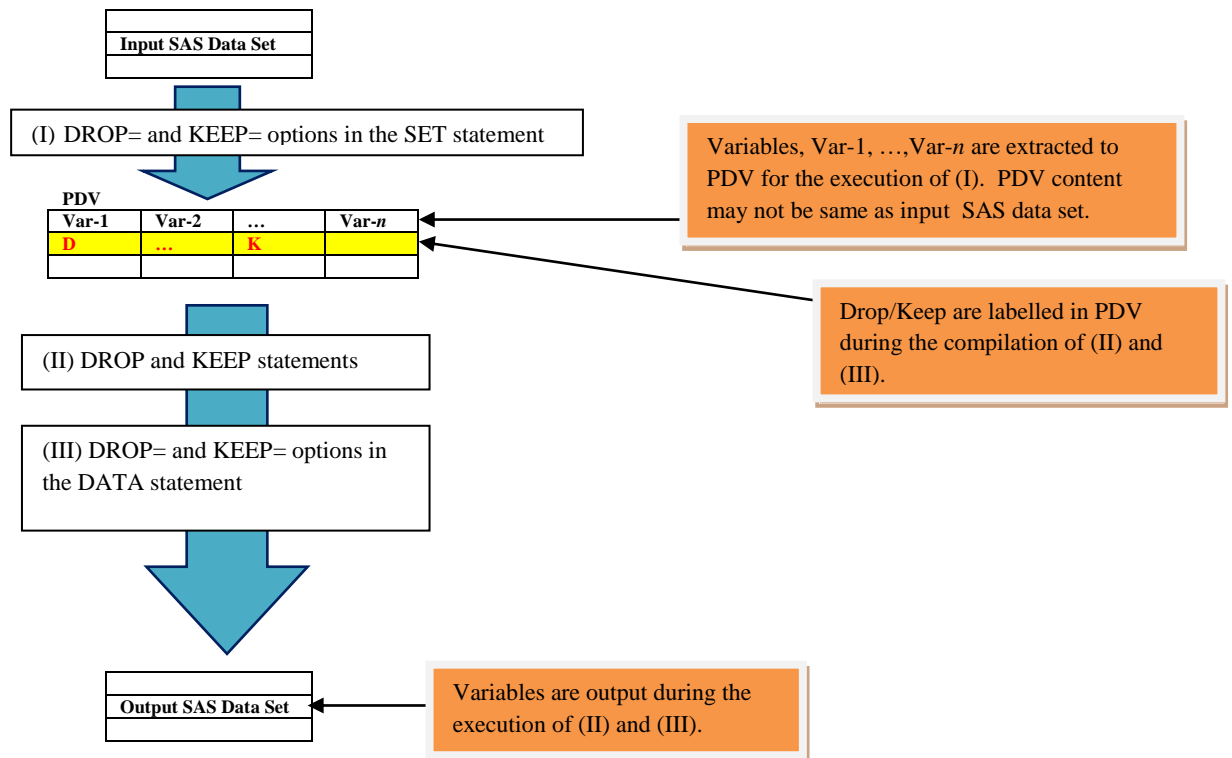


- Using KEEP= options on an output data set:



-  Difference among these KEEP/DROP statement, KEEP=/DROP= options in SET statement and DATA statement

- The following diagram summarizes the process of PDV for the commands in the SAS program.



**Ex. 5.2:** Submit the following SAS program. It is intended to store the records having “visit\_country” as Singapore in Work.Singapore and the records having “visit\_country” as Australia in Work.Australia. The remaining is stored in Work.others.

```

*p5ex2.sas;
data visit;
  length Last_Name $20 First_Name $20 Gender $1 Resid_District $20
  Home_country $20 Project_Field $20 Visit_country $20;
  infile 'd:\SAS_datasets\AMAYr3visit.csv' dlm=',';
  input Uni_no First_name $ Last_name $ Gender $ Telephone
  Resid_District $ Home_country $ Project_Field $ Visit_country $;
run;

data singapore australia(drop=home_country) others;
  set visit(drop=visit_country gender telephone);
  if UPCASE(visit_country)='SINGAPORE' then output singapore;
  else if UPCASE(visit_country)='AUSTRALIA' then output australia;
  else output others;
run;
  
```

- Can the LENGTH statement be omitted in the program?
- How many observations do the data sets (Work.singapore, Work.australia, Work.others) have?
- Correct the program so that it fits for the purpose stated.

```
data singapore australia(drop=home_country)  others;
  set visit(drop=visit_country gender telephone);

  if UPCASE(visit_country)='SINGAPORE' then output singapore;
  else if UPCASE(visit_country)='AUSTRALIA' then output australia;
  else output others;

run;
```

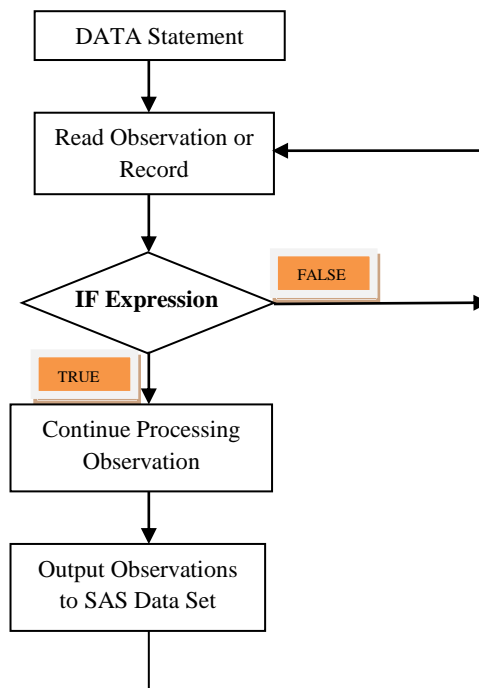
## 5.2 Subsetting observations

### Subsetting IF statement

- Syntax

```
IF Expression;
```

- Processing the subsetting IF statement



When a DATA step is processed, it will process in the way as the above flowchart. Once the “IF Expression” is encountered, it will continue to process the statement(s) below the “IF Expression” when a true is processed. When a **false** is processed, the content of the PDV will **NOT output** to the output data set.

- Electricity data revisit:

```
*p5_example1.sas;
libname xls "D:\SAS_Datasets\electricity.xls";
data Years1;
  set xls.'Year1-4$'n;
  if AvTemp ne . ;
  /* Avcost=average cost per day*/
  Avcost=cost/Deg_Days;
run;
```

- There are 24 observations in the Sheet ‘Year1-4’ of Electricity.xls and the variable, AVTEMP, average temperature is missing for 2 observations.
- Only those 22 observations of AVTEMP with non-missing values are output to data set, Work.Years1. And the average cost is evaluated for these 22 observations.
- The following subsetting IF statement and IF-THEN DELETE statement have the same effect:

```
IF Expression1;  
IF NOT Expression1 THEN DELETE;
```

The DELETE statement stops processing the current observations.

#### WHERE statement

- Syntax

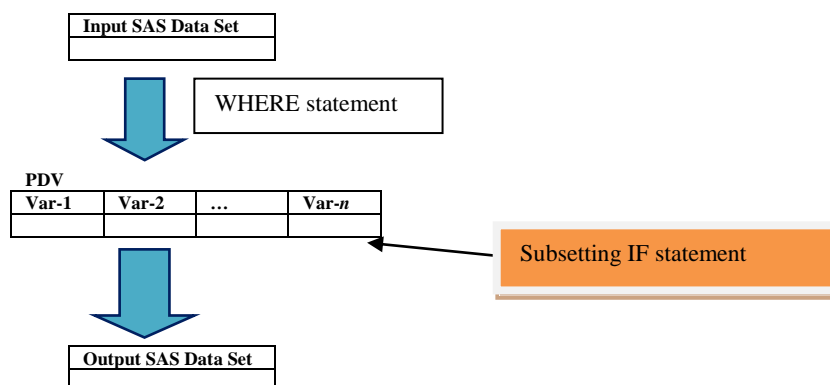
```
WHERE Expression;
```

- Electricity data revisited:

```
*p5_example1.sas;  
libname xls "D:\SAS_Datasets\electricity.xls";  
data Years1w;  
  set xls.'Year1-4$'n;  
  where AvTemp ne .;  
  /* Avcost=average cost per day*/  
  Avcost=cost/Deg_Days;  
run;
```

- This file generates the same output as the program using subsetting IF statement.

#### Difference between subsetting IF statement and WHERE statement in PDV



### 5.3 Create multiple observations in the data set

#### Use of explicit OUTPUT

- Syntax

```
OUTPUT;
```

Output the current observation in PDV to the output SAS data set.

Note that “implicit output;” is not imposed when “output;” statement is found one line before the “run;” statement. See the following example.

- Example: AMA Graduate

- 5 students graduate records:

Record	Uni_ID	First_Name	Last_Name	Gender	GradYr
1	90010101	Peter	Smith	M	08
2	89241253	Zhian	Yang	M	11
3	91021201	Chi Wah	Chan	F	95
4	07012345	Amy	Yip	F	11
5	86012345	Rose	Yip	F	89

- SAS program to produce these five records:

```
*p5_example2.sas';
data AMA_grad;
  Uni_ID=90010101;
  First_Name='Peter';
  Last_Name='Smith';
  Gender='M';
  GradYr=08;
  Output;
  Uni_ID=89241253;
  First_Name='Zhian';
  Last_Name='Yang';
  Gender='M';
  GradYr=11;
  Output;
  Uni_ID=91021201;
  First_Name='Chi Wah';
  Last_Name='Chan';
  Gender='F';
  GradYr=95;
  Output;
  Uni_ID=07012345;
  First_Name='Amy';
  Last_Name='Yip';
  Gender='F';
  GradYr=11;
  Output;
  Uni_ID=86012345;
  First_Name='Rose';
  Last_Name='Yip';
  Gender='F';
  GradYr=89;
  Output;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

 **Ex. 5.3:** Submit the above program and answer the questions.

- (a) What is the stored value of First\_Name for the record with Uni\_ID=91021201? What is the length of First\_Name and why? Suggest a method to correct the error.

```
data AMA_grad1;
```

```
Run;
```

- (b) Correct all errors found if any in Part (a). Write a SAS program to print the 5 records. Output the graduate year using 4 digits.

```
data Print_grad;  
  set AMA_grad1;
```

- (c) With reference to the sample program (p4example1.sas) on page 4.3, write a SAS program to store the records for graduates 1995, 2008 and 2011 from AMA graduates data set into data sets of name grad95, grad08 and grad11 respectively. Store the remaining records to grad\_others.

#### 5.4 Controlling which observations to be read

##### FIRSTOBS= option

- Syntax

```
SAS-data-set (FIRSTOBS=n) ;
```

- It specifies which observation is the first one to be read for process. By default, SAS will process to the end of the input data set.
- It is used in statements where data set is listed, like SET statement, PROC statement.

##### OBS=option

- Syntax

```
SAS-data-set (OBS=n) ;
```

- It specifies which observation is the last one to be read for process. By default, SAS will process from the beginning of the input data set.
- It is used in statements where data set is listed, like SET statement, PROC statement.

##### Combined options

- Syntax

```
SAS-data-set (FIRSTOBS=m OBS=n) ;
```

- It specifies which observation is the first one (*m*-th observation) and last one (*n*-th observation) to be read for process.
- It is used in statements where data set is listed, like SET statement, PROC statement.

- AMA Graduate Record data revisit
  - Printing of Observations 2 to 4.

```
*p5_example2.sas;
proc print data=Print_grad (Firstobs=2 Obs=4);
format Uni_ID Z8.0 GradYr Z4.0;
run;
```

The output:

Obs	First_ Name	Last_ Name	Uni_ID	Gender	Grad Yr
2	Zhian	Yang	89241253	M	2011
3	Chi Wah	Chan	91021201	F	1995
4	Amy	Yip	07012345	F	2011

- Extracting of first four observations into a data set called NEW.

```
*p5_example2.sas;
data New;
set AMA_grad1 (OBS=4);
run;
```

#### In INFILE statement

- Syntax

```
Infile 'raw-data-file' <FIRSTOBS=m OBS=n>;
```

- No parentheses are required for the FIRSTOBS= and OBS= options.
- Either option can be used.

- An example: Graduate 1993 data revisit

```
*p5_grad93.sas;
Data work.grad93m;
Infile 'd:\SAS_datasets\AMAggrad93.csv' dlm=',' firstobs=2 obs=4;
Length First_Name $ 20 Last_Name $20 Gender $1
Job_Title $25 Country $25 Phone $ 20;
Input Uni_ID First_Name $ Last_Name $ Gender $ DOB:DDMMYY.
Current_Salary: Dollar. Job_Title $ Country $ Phone $;
Run;
```

- Work.grad93m read the 2<sup>nd</sup> observation up to the 4<sup>th</sup> observation according to the *firstobs=* and *obs=* options.
- These two options can also be applied to other type of raw data file.

```
*p5_grad93.sas;
Data work.grad93m;
Infile 'd:\SAS_datasets\AMAggrad93.DAT' firstobs=2 ;
Length First_Name $ 20 Last_Name $20 Gender $1
Job_Title $25 Country $25 Phone $ 20;
Input @1 Uni_ID 8. @9 First_Name $7. @16 Last_Name $6. @22 Gender $1.
@24 DOB:DDMMYY10.
@34 Current_Salary 7.0 @41 Job_Title $23. @64 Country $11. @75
Phone $14.;
Run;
```



### Adding a WHERE statement

- Execution order:
  - WHERE statement will be executed first
  - The FIRSTOBS= and OBS= options are applied to **the resulting observations**.
- An example

```

Data work.grad93;
  Infile 'd:\SAS_datasets\AMAGrad93.csv' dlm=',';
  Length First_Name $ 20 Last_Name $20 Gender $1
         Job_Title $25 Country $25 Phone $ 20;
  Input Uni_ID First_Name $ Last_Name $ Gender $ DOB:DDMMYY.
        Current_Salary: Dollar. Job_Title $ Country $ Phone $;

Run;
proc print data=work.grad93(obs=3);
where gender='M';
run;
    
```

Work.grad93

	Uni_ID	First_Name	Last_Name	Gender	DOB	Current_Salary	Job_Title	Country	Phone	Join_Date
1	90010101	Peter	Smith	M	1/7/1965	600000	Professor	Austria	(00431)58000	7/9/1997
2	89241253	Zhian	Yang	M	15/8/1964	750000	Associate Professor	macau	(00853)123456	1/9/1994
	91021201	Chi Wah	Chan	F	6/8/1973	500000	Consultant	Hong Kong	(00852)25698561	1/8/1993
	89012345	Amy	Yip	W	5/6/1972	1000000	Government statistician	Hong KONG	(00852)29090612	1/8/1938
3	90893208	Tony	Brown	M	8/6/1972	555000	Lecturer	Switzerland	(004441)1234567	7/9/1997
	90893208	Tony	Chung	M	9/9/1974	34567	Statistical Officer	HONG KONG	(00852)37090612	unknown
	92304567	Rebecca	Wicker	G	8/6/1972	-70000		Hong Kong	123	79/9/1997

- First step: Subsetting WHERE statement is applied and this results with the yellow highlighted observations
- Second step: Process FIRSTOBS= and OBS= options. OBS=3 represents the last observation is three. Therefore, only those yellow marked records with labels will be printed because the Procedure Print will print up to the 3<sup>rd</sup> observation and the 4<sup>th</sup> yellow highlighted observation is omitted.



### Ex. 5.4: AMA Graduate 1993 records revisit - Referring to last program:

```

Data work.grad93;
  Infile 'd:\SAS_datasets\AMAGrad93.csv' dlm=',';
  Length First_Name $ 20 Last_Name $20 Gender $1
         Job_Title $25 Country $25 Phone $ 20;
  Input Uni_ID First_Name $ Last_Name $ Gender $ DOB:DDMMYY.
        Current_Salary: Dollar. Job_Title $ Country $ Phone $;

Run;

proc print data=work.grad93(obs=3);
where gender='M';
run;
    
```

- If “OBS=3” is replaced with “FIRSTOBS=3” in the above Procedure Print, whose records are output?
- If “gender=’M’” is replaced with “gender=’F’” in the above Procedure Print, which university ID records are output?
- If “OBS=3” is replaced with “FIRSTOBS=3” in addition to the change in part (b), which university ID records are output?

## 5.5 Create multiple data sets

### Syntax

```
DATA <Output-SAS-data-set-1>< Output-SAS-data-set-2 ...Output-SAS-data-set-n>;  
Statement(s);  
OUTPUT<Output-SAS-data-set-1> ...<Output-SAS-data-set-n>;  
.....  
RUN;
```

For the above OUTPUT syntax, it direct output to all Output-SAS-data-sets listed in the OUTPUT statement. IF-THEN-ELSE statement is used for directing the output conditionally. (See part (c) of the above exercise). In the case of no argument, every observation is written to all SAS data sets listed in the DATA statement.

### A SELECT Group can also be used for the multiple data sets.

#### - Syntax

```
SELECT <{select-expression}>;  
    WHEN (value-1<...,value-n>) statement*;  
< WHEN .....  
    WHEN (value-1<...,value-n>) statement*;>  
<OTHERWISE statement*;>  
END;
```

- \*If more than one statement is required, use a *SELECT group* for multiple statements.
- The *select-expression* specifies any SAS expression to a single value. It is often a variable name within the SAS program or in a SAS data set or value of a SAS function.
- When the *select-expression* is omitted, the values in the WHEN statements are changed to expressions.

```
SELECT ;  
    WHEN (expression-1) statement*;  
< WHEN .....  
    WHEN (expression-n) statement*;>  
<OTHERWISE statement*;>  
END;
```

SAS proceed the WHEN statement sequentially. When the expression is true, the associated statements are executed. If it is false, SAS will proceed to the next WHEN statement. If all WHEN expressions are false, the statement(s) followed by otherwise statement executes.

- The OTHERWISE statement is optional but omitting it may result an error when all WHEN conditions are false. To prevent SAS from issuing an error message, use

OTHERWISE;

#### - Syntax for creating multiple data sets

```
DATA Output-SAS-data-set-1 <... Output-SAS-data-set-n>;  
SET Input-SAS-data-set;  
SELECT <(select-expression)>;  
    WHEN (value-1<...,value-n>) OUTPUT <Output-SAS-data-set-1 ...  
        Output-SAS-data-set-n>;  
<WHEN .....  
    WHEN (value-1<...,value-n>) OUTPUT <Output-SAS-data-set-1 ...  
        Output-SAS-data-set-n>;>  
<OTHERWISE OUTPUT <Output-SAS-data-set-1 ...  
        Output-SAS-data-set-n>;>  
END;  
RUN;
```

- AMA Graduate Records Revisit

```
*p5_example2.sas';
data grad95 grad08 grad11 grad_others;
  set AMA grad1;
  select (GradYr);
    when (95) output grad95;
    when (08) output grad08;
    when (11) output grad11;
    otherwise output grad_others;
  end;
run;
```

The yellow portion statements are one SELECT group which generates the same outcomes as using IF-THEN-ELSE statements.

The yellow portion can be re-written as

```
*p5_example2.sas';
select;
  when (GradYr=95) output grad95;
  when (GradYr=08) output grad08;
  when (GradYr=11) output grad11;
  otherwise output grad_others;
end;
```

## 5.6 WHERE statement versus Subsetting IF statement

Step and Usage	WHERE statement	Subsetting IF statement
PROC step	Yes	No
DATA step (source of variable)		
INPUT statement	No	Yes
Assignment statement	No	Yes
SET statement (single data set)	Yes	Yes
SET/MERGE* statement (multiple data sets)		
Variable in ALL data sets	Yes	Yes
Variable not in ALL data sets	No	Yes

\*Merge statement is used to merge two SAS data sets and will be discussed in Chapter 6.