SAS Tutorial

PRESENTED BY : SHASHI KUMAR


python
sas
VS
R

What is SAS & Why SAS ??

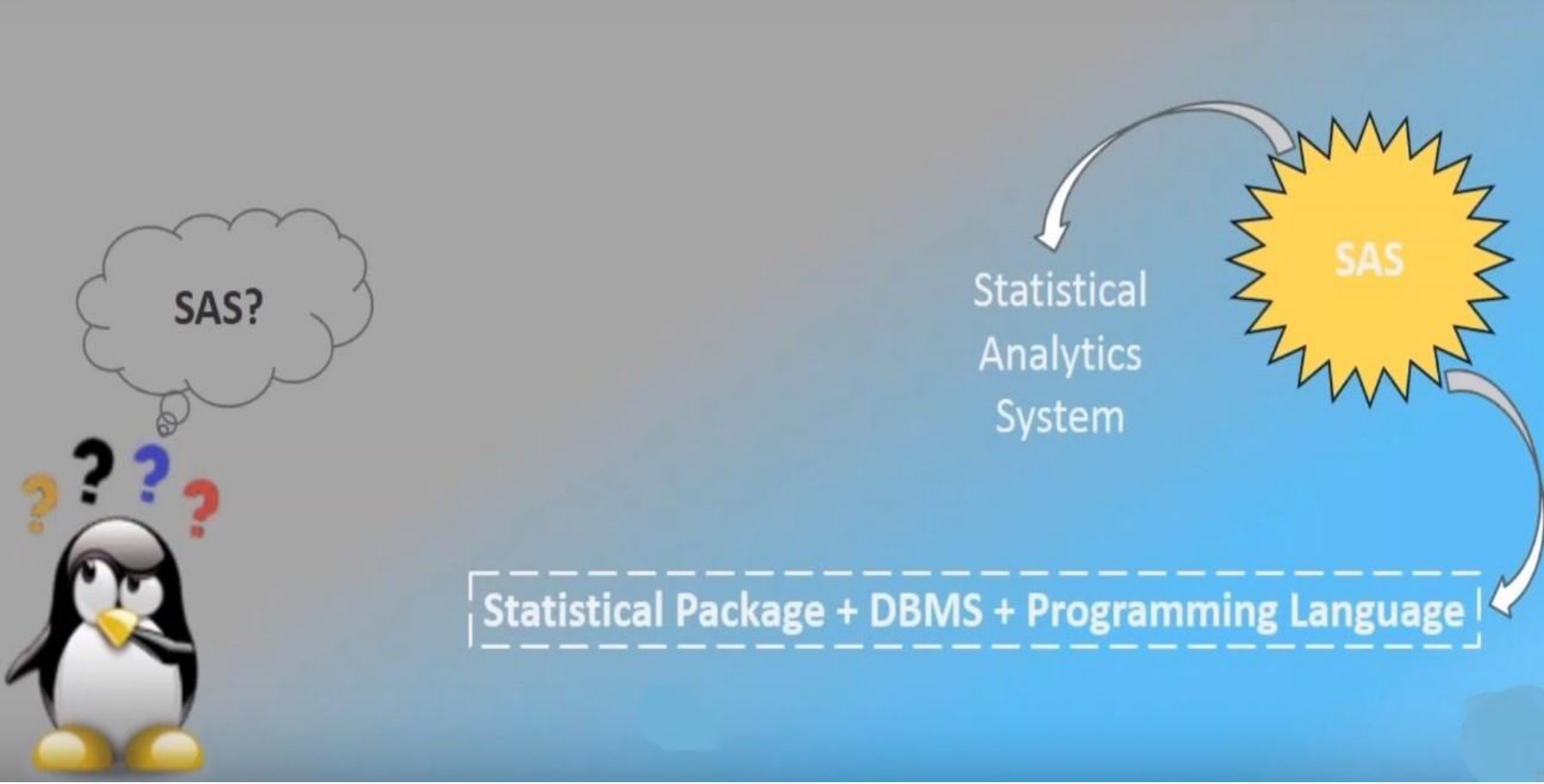
SAS ("Statistical Analysis System") is a software suite developed by SAS Institute for advanced analytics, multivariate analyses, business intelligence, data management, and predictive analytics.

Here is a brief description about the 3 ecosystems:

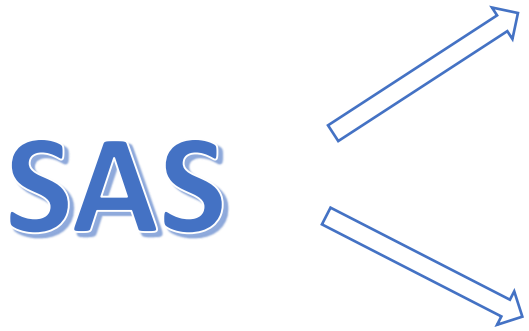
SAS: SAS has been the undisputed market leader in commercial analytics space. The software offers huge array of statistical functions, has good GUI (Enterprise Guide & Miner) for people to learn quickly and provides awesome technical support. However, it ends up being the most expensive option and currently being updated with latest trends like SAS Viya(R &Python).

R: R is the Open source counterpart of SAS, which has traditionally been used in academics and research. Because of its open source nature, latest techniques get released quickly. There is a lot of documentation available over the internet and it is a very cost-effective option.

Python: With origination as an open source scripting language, Python usage has grown over time. Today, it sports libraries (numpy, scipy and matplotlib) and functions for almost any statistical operation / model building you may want to do. Since introduction of pandas, it has become very strong in operations on structured data.



Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE



Program

BASE SAS

Advance SAS

Graphical User Interface (GUI)

Data Integration (DI)

Visual Analytics (VA)

Enterprise guide(EG)

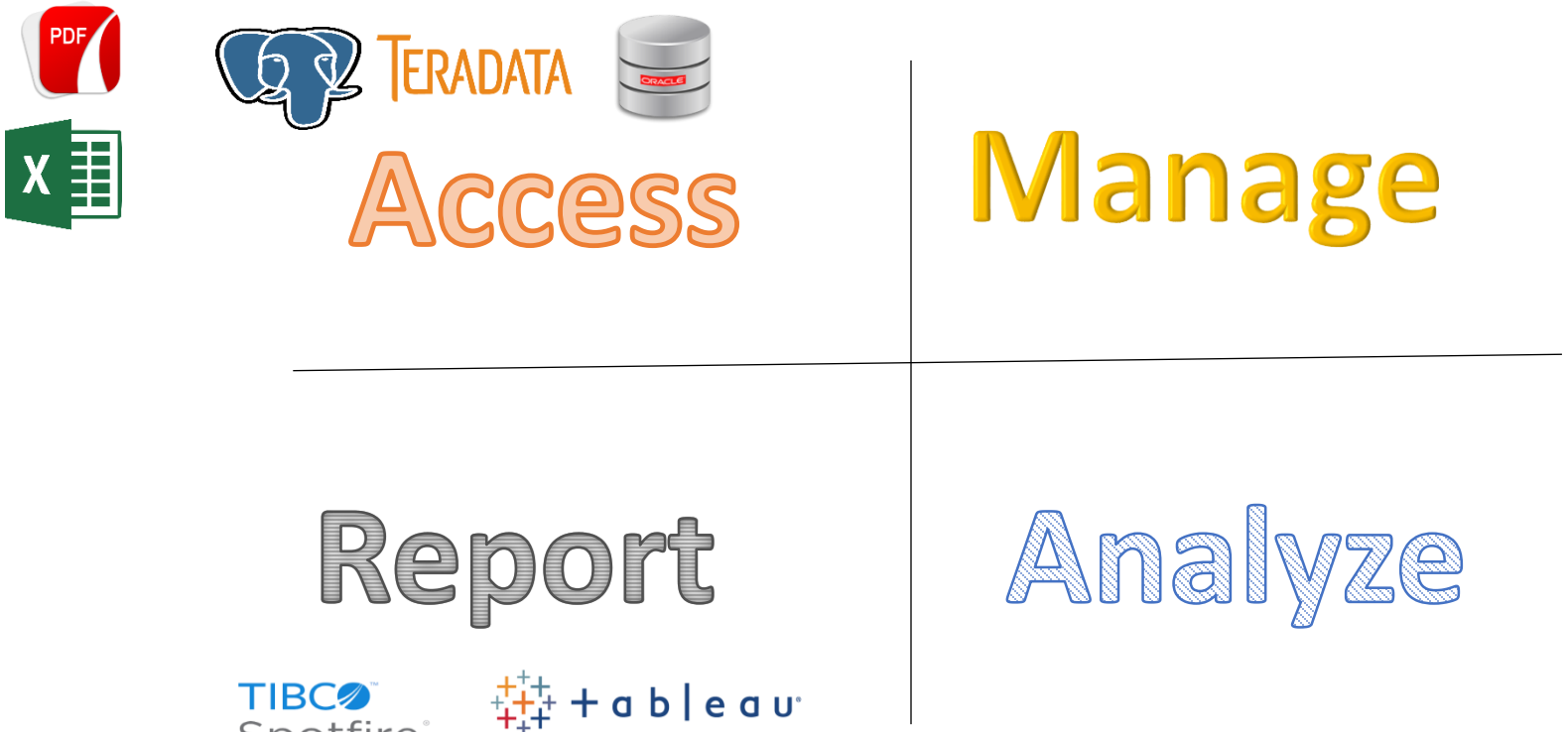
Customer Intelligence (CI)

Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

Foundation Of SAS

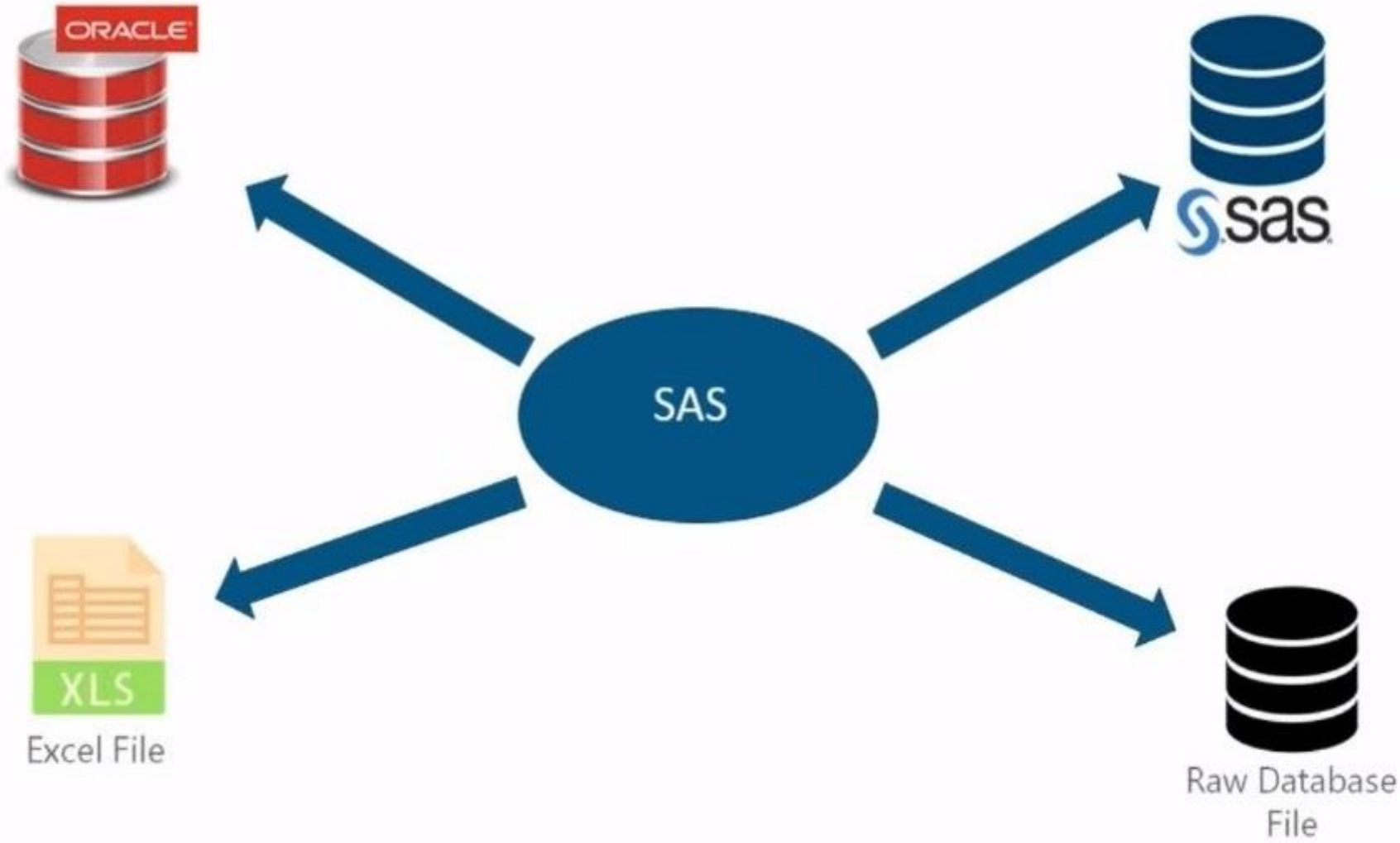
SAS :-

It is a highly flexible and integrated software environment that is used to access, manipulate, manage, analyze and report of data.



Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

1. Access
2. Manage
3. Analyze
4. Present / *Report*



1 Access

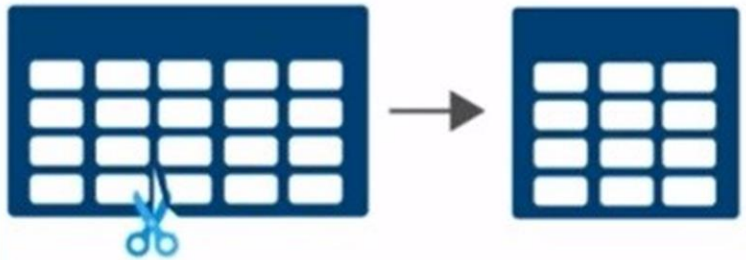
2 **Manage**

3 Analyze

4 Present

SAS gives you excellent **data management** capabilities

1)Subset Data



2)Create Variables

3)Clean & Validate Data

1. Access
2. Manage
3. **Analyze**
4. Present

After Data Management the next step is **data analysis :**

Frequency or Mean calculation

Regression and Forecasting

SAS is the gold standard for statistical analysis.

Once you have analyzed data you can present it better with SAS

1. Access
2. Manage
3. Analyze
4. **Present**

/ Report

SAS Applications

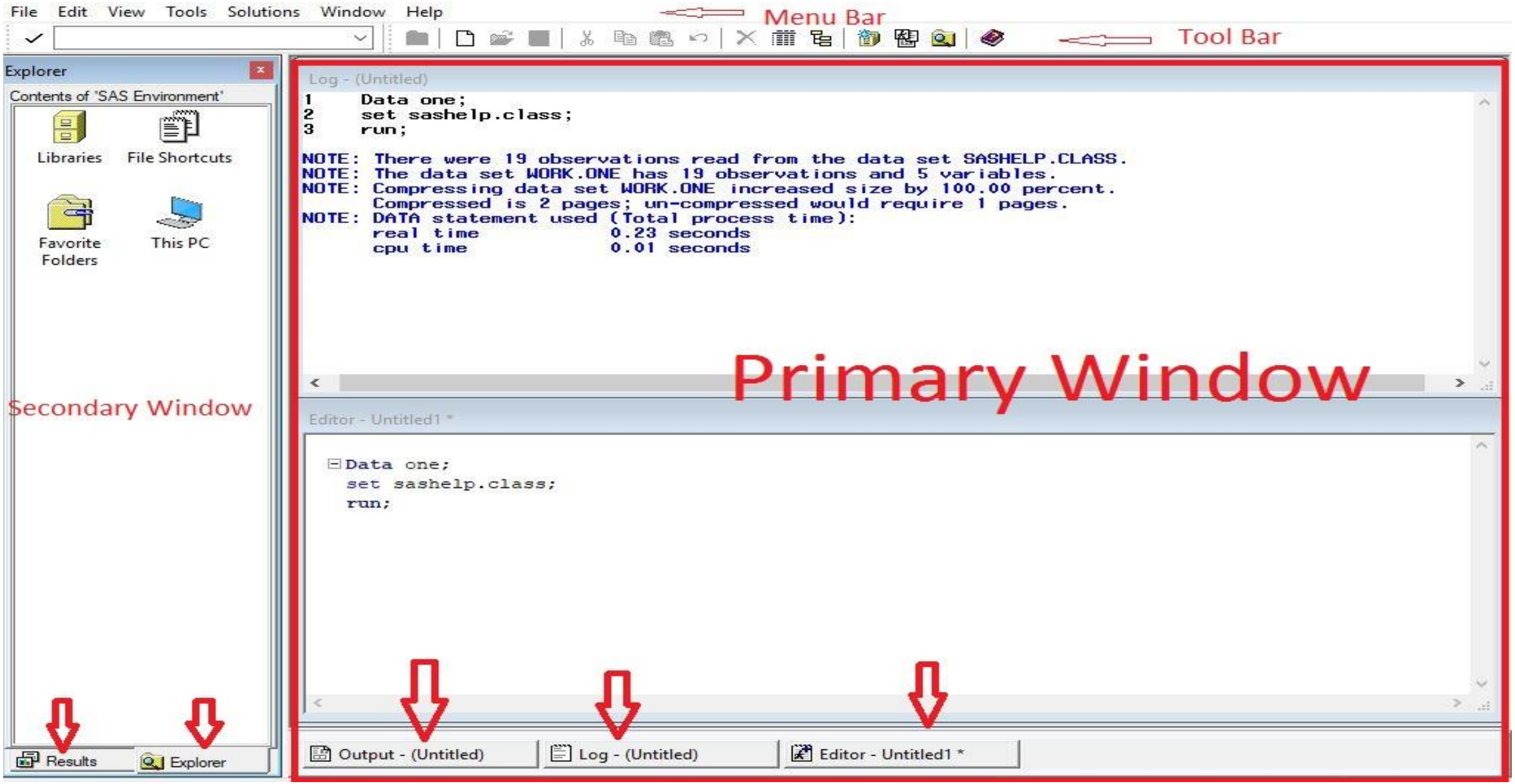
1)Stock Prediction

2)Create Safe Drugs

3)Fight Fraud

4)Optimize Workflow

SAS Session



Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

SAS Session

Primary Window

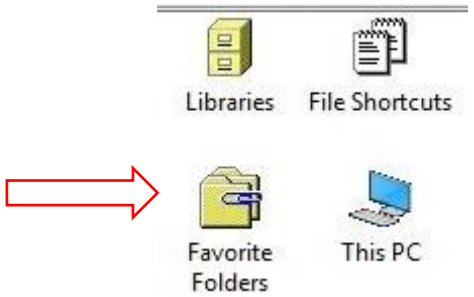
1. Outputs: Contain the report of procedure that have submitted and executed
2. Log: Provide Information about SAS program execution.

   a. Note : Blue color
   Numbers of observation
   Data set names
   b. Warning: Green Color
   Execution Continue
   c. Error: Red Color
   Depend on error it will stop or continue the execution

3. Editor: The Place where SAS program is written, edited, submitted the program for execution.

Secondary Window

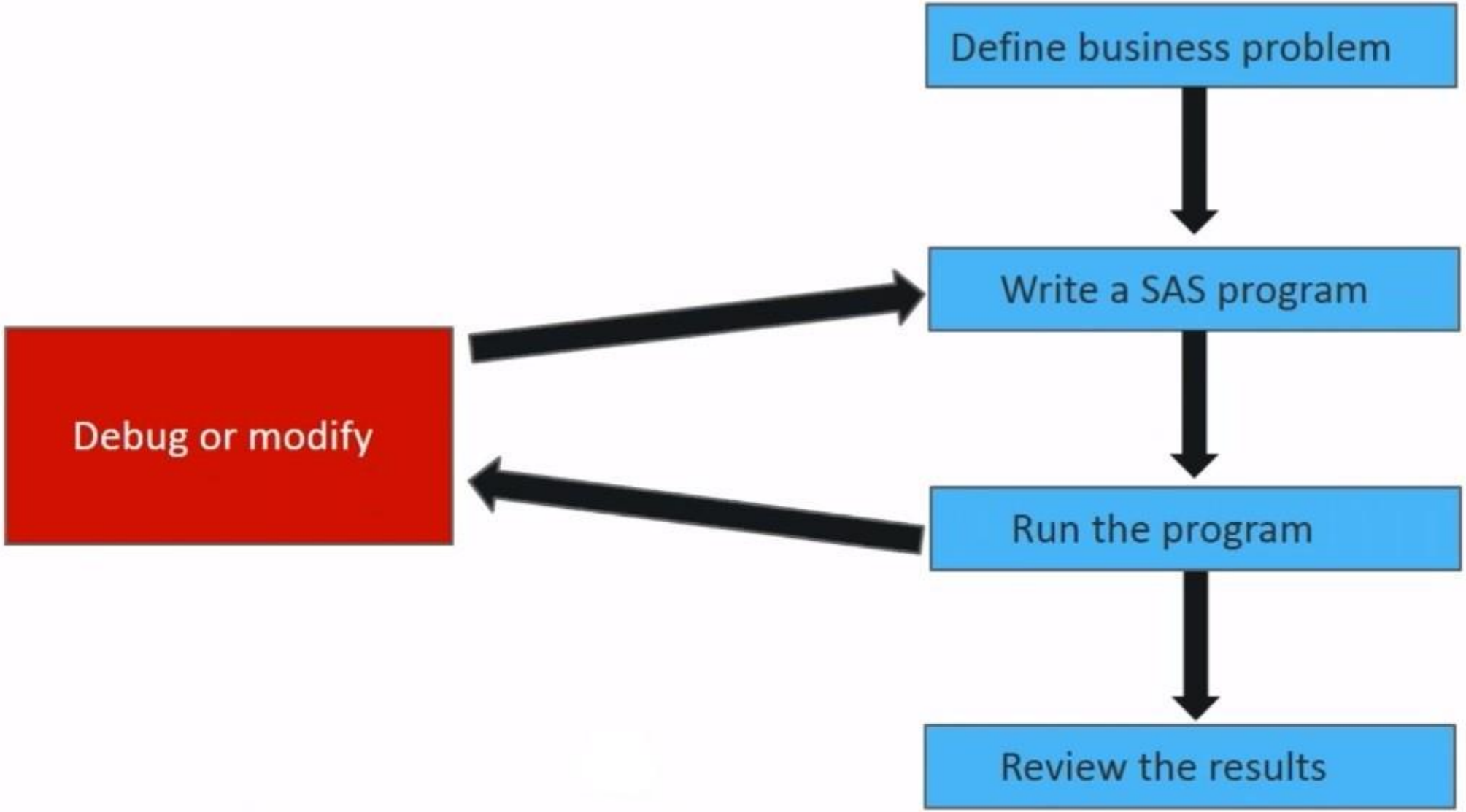
1. Result : It contains list of procedure which are submitted and executed successfully.
2. Explorer : Provide easy navigation to SAS library icon ,window system, my computer etc.



SAS Programming Language

Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

SAS Programming Process



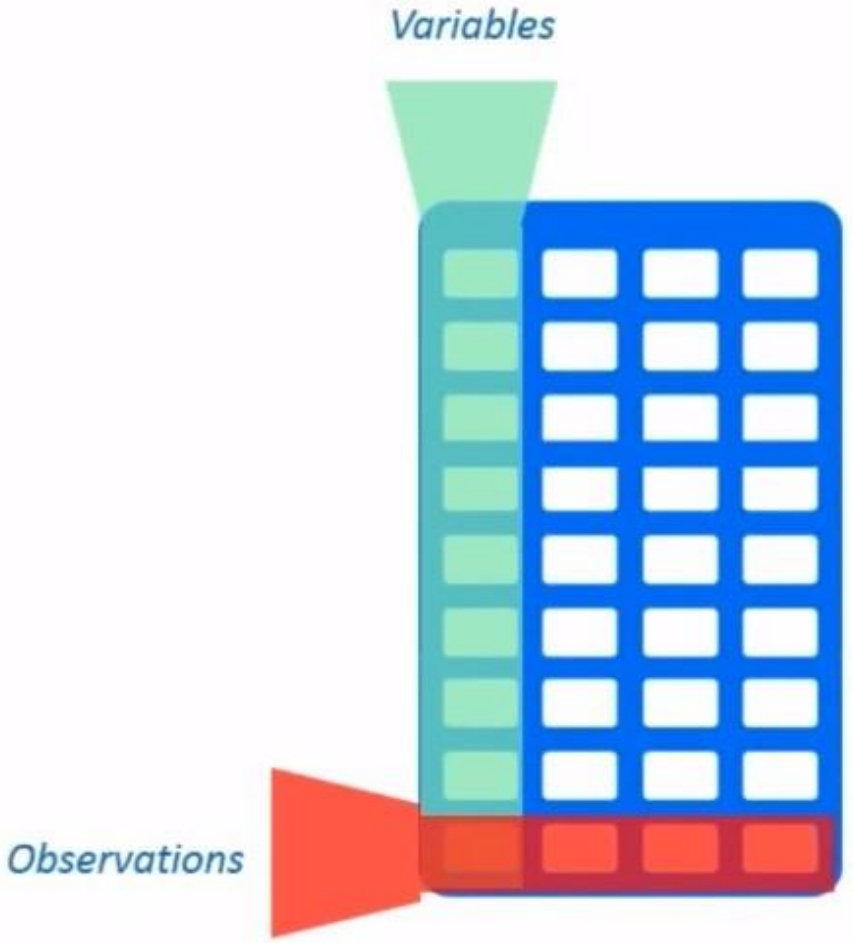
SAS Data

Data is central to every data set.

- In SAS Data is in tabular form
- Variables occupy the columns
- Observations occupy the rows

Data types:

- Numeric
- Characters



SAS Program Structure

SAS programming is based on two building blocks:

1)DATA Steps

DATA steps create or modify SAS data sets. Using DATA steps you can:

- Add data to a data set
- Compute values of variables
- Create new data sets (by sub-setting, merging)

SAS Program Structure

SAS programming is based on two building blocks:

2)PROC Steps

PROC steps analyse and process SAS data sets. Using PROC steps you can:

- Print a report
- Produce descriptive analysis
- Create a tabular report
- Produce plots and charts

Topic: Data Step , Boundary, Statement

```
****************************************
/* What is Statement  */


/* Statement beigns with reserved keyword and ends with ; */;

data a;  * this is also a statement ;
set zzz; * this is also  a stement ;
run;     * this is also a statemnent;

************************************;
```

```
/*Multiple SAS statements is written in single row */
Data one;set sashelp.class;run;
/*A single SAS statement can be written in multiple row*/
Data
one;
set
sashelp.class;run;
```

The **DATA step** consists of a group of **SAS** statements that begins with a DATA statement. The DATA statement begins the process of building a **SAS** data set and names the data set. The statements that make up the **DATA step** are compiled, and the syntax is checked.

1. **SAS Statement** : A SAS Statement begin with SAS identifying Keyword and end with semi column (**;**).
   Properties:-
   - 1.1 A Single SAS statement can be written in multiple row.
   - 1.2 Multiple SAS statement is written in single row.
   - 1.3 One or more blank separated the word.
2. **Step Boundary** : Program ends with SAS identifying Keywords e.g. **run**, **quit** and **begin of new SAS program.**
3. **SAS Step** : It is a combination of SAS statements ,the SAS step begin or start with identifying keyword i.e. **data** or **proc** and end with step boundary.

Topic: Data Step , Boundary, Statement

```
/* Sample Code */
data a;
set sashelp.class;
run;

*******************************************;

/*Data step  begins with a  keyword "data" and this step ends with either of 3 ways (run; or beginning of new pgm or quit; )   */
 /* First method  */

 data a;
 set sashelp.class;
 run;


 ******************************;

 /* Second Method; */

 data b;
 set sashelp.class; *this pgm works without run statement because after this there is a new beginning of pgm;

 data c;
 set sashelp.class;
 run;

 /* Third Method  */

 /* Quit statement is used with procs only */

 proc print data=a;
 quit;
```

Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

Exercise :

```
*%%%%%%%%%% Create SAS dataset using Set statement %%%%%;
Data  one;
Set  sashelp.class;
Run;

*%%%%%%%%%% Create SAS dataset using datalines %%%%%;
Data two;
Infile datalines dlm=" ";
Input ID NAMES $;
Datalines;
1 Shashi
2 Ravi
3 Mohan
;
run;
```

Topic: SAS Variable

Numeric	Character
It can hold 0-9,integer number, decimal number	It can hold any character values, such as letter or number, special character and " " (blank)
Right align	Left align
Missing Value/Blank assign as **.** (**dot**)	Missing Value/Blank assign as " " (**Space**)
Default length is **8 bytes**	Default length is **8 bytes**
16-17 digit number when 8 bytes	8 bytes hold 8 character
Minimum length is **3 bytes**	Minimum length is **1 bytes**
Maximum length is infinite (depend on RAM Size)	Maximum length is 32767

Topic: SAS Naming Convention for Variable and Data set

1. Name can hold maximum **32** character.
2. Name should begin with _ (under score) or letter and seconds onwards letter can be _ (under score),letter or numeric.
3. Special Character not permitted except _ (under score).
4. SAS name is case insensitive ,it may be upper, lower or prop case.

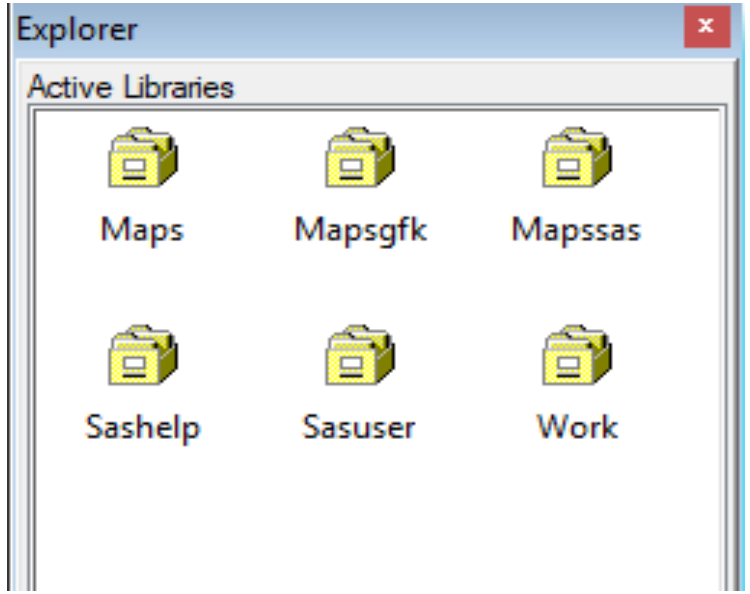
test	✓
test123	✓
_test	✓
@test	✗
4test	✗
Test&	✗
TEST	✓
_12test	✓

Library :

- SAS library is simply a collection of SAS files that are stored in the same folder or directory on your computer. Other files can be stored in the same folder or directory, but only the files that have SAS file extensions are recognized as part of the SAS library.
- Depending upon on your need SAS library is of two types :

Work as Temp

All content deleted after SAS session



Sashelp or lib ref are permanent

Content retained even after SAS session

Dataset | View | Catalog | SAS pgm file

Library : Temporary to permanent

Syntax:

Libname <libref> <engine> path;

i.e. Libname sk “D:\Users\703215742\Desktop\SAS CLASS”;

This LIBNAME statement specifies ***sk*** as a reference to a SAS library. The EXCEL engine specifies the engine that supports the connection to the file type .XLSX.

Libname sk EXCEL “D:\Users\703215742\Desktop\SAS CLASS\file.xlsx”;

Libname sk “D:\Users\703215742\Desktop\SAS CLASS\file.xlsx”;

Access Tera Data in SAS:

libname sk teradata user="userid" password="password" mode=teradata server="servername“ connection=global dbmstemp=yes;

<libref> : - Naming Convention

1. Name can hold maximum **8** character.
2. Name should begin with _ (under score) or letter and seconds onwards letter can be _ (under score),letter or numeric.
3. Special Character not permitted except **_** (under score).
4. SAS name is case insensitive ,it may be upper, lower or prop case.

Library : Temporary to Permanent

Temporary to Permanent Library

```
Data one;
Set sashelp.class;
Run;
```

Dataset **one** is stored in temporary library "**Work**".
(All content deleted after SAS session)

```
Libname sk "D:\Users\7032xxxxx\Desktop\SAS CLASS";

Data sk.two;
Set sashelp.class;
Run;
```

Dataset **two** is stored in Permanent library "**sk**".
(Content retained even after SAS session)

Proc Print :-

Syntex:-

```
proc print data =input data <option>;
Var var1 var2…varn;
Id var1 var2 …varn;
run;
```

Var : It define the variables and observation in o/p window.
Id: It suppress the observation column and id variable comes the first variable in o/p window.

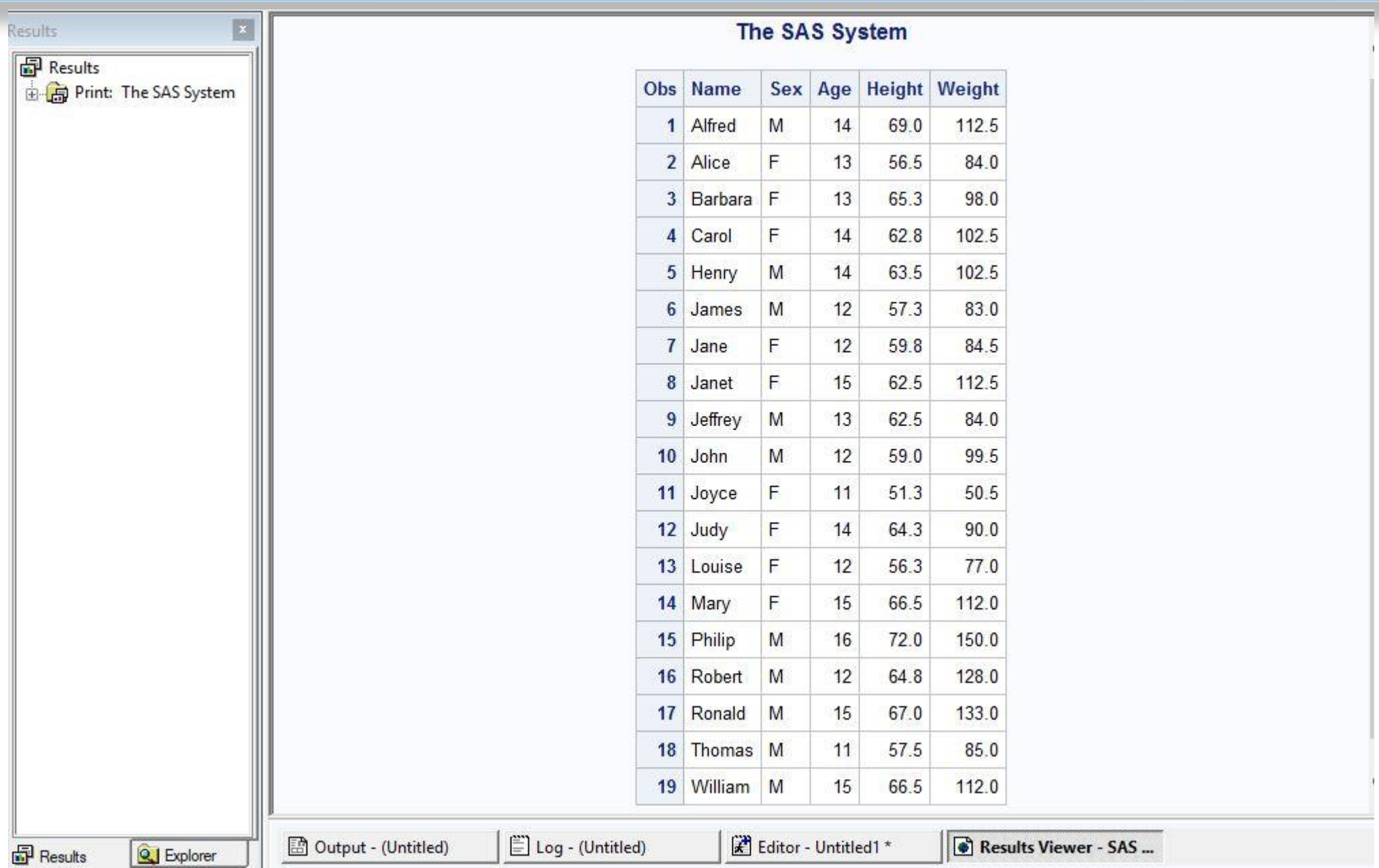
Options:-
Noobs:- It suppress the observation column in the proc print o/p.
Double:- It provides space between the observation.
Obs=n:- It give the first n observation from the dataset.

Examples:-

```
proc print data =sashelp.class;run;
proc print data =sashelp.class; var age sex;run;
proc print data =sashelp.class; id age;run;
proc print data =sashelp.class double ;run;
proc print data =sashelp.class (obs=10);run;
```

Proc Print :-

```
proc print data =sashelp.class;run;
```



Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Proc Contents:-

Syntax:-

proc contents data=input data <option>;**run**;

It display description portion of dataset by default by show the result in three parts.

1. Attributes
2. Engine/Host
3. List of Variables

Examples:-

proc contents data=sashelp.class; **run**;

proc contents data=sashelp.class varnum; **run**;

proc contents data=sashelp._all_; **run**;

proc contents data=sashelp.class nods; **run**;

Proc Contents:-

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	09/06/2017 21:55:32	Observation Length	40
Last Modified	09/06/2017 21:55:32	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_64		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1632
Obs in First Data Page	19
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp\class.sas7bdat
Release Created	9.0401M5
Host Created	X64_SR12R2

Owner Name	BUILTIN\Administrators
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Exercise:-

1. **Create Permanent Library**

```
Data one;
Set sashelp.class;
Run;
```

```
Libname sk "D:\Users\7032xxxxx\Desktop\SAS CLASS";

Data sk.two;
Set sashelp.class;
Run;
```

2. **proc print** data =sashelp.class;**run**;
3. **proc contents** data=sashelp.class; **run**;

PDV : Program Data Vector

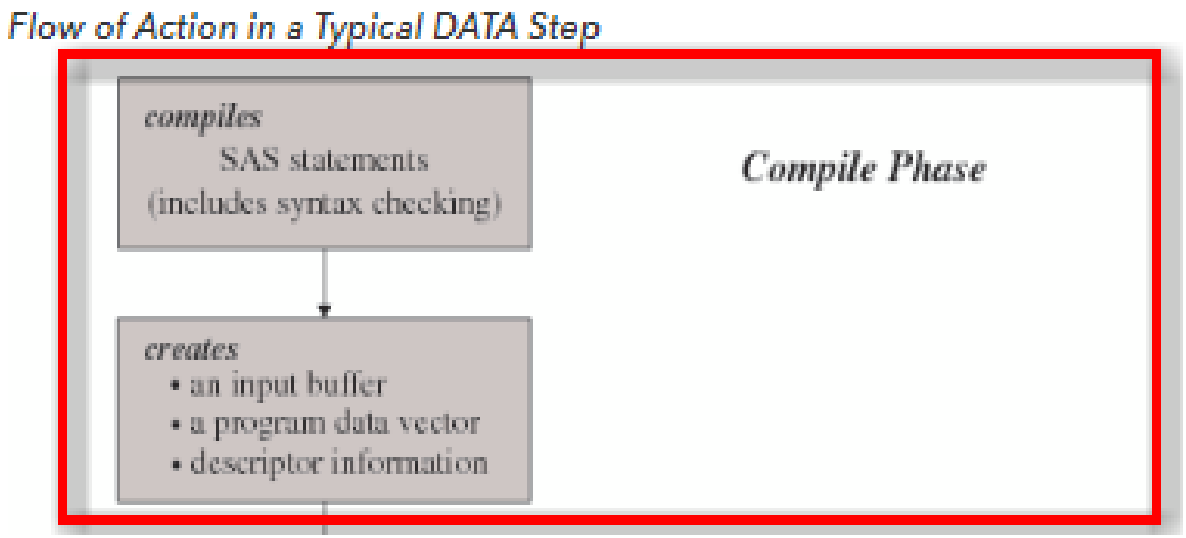
Before Actually moving to the PDV section let us try to explorer how SAS actually its data/steps in a sequence.

SAS initiate its code into two parts :

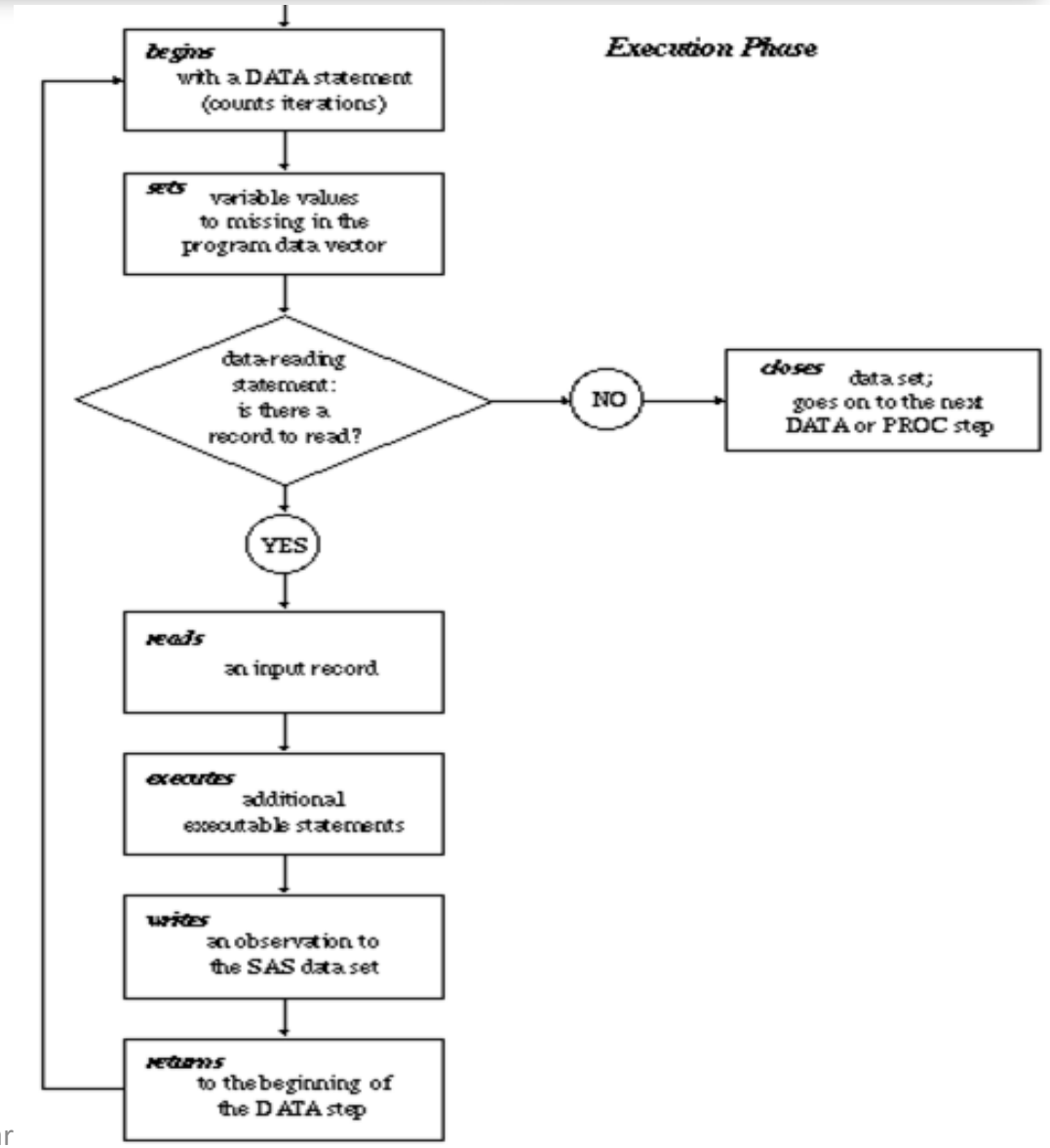
1. Compilation Phase
2. Execution Phase

- In the compilation phase SAS checks the syntax of the submitted code and if there is a syntax error then SAS stops the further process and at the end of the compilation phase (i.e. after checking syntax and good to go) SAS creates the descriptor portion of the dataset.

Compilation Phase :



1. Syntax Error
2. Create PDV
3. Create descriptor portion



Compilation Phase:

When you submit a DATA step for execution, SAS checks the syntax of the SAS statements and compiles them, that is, automatically translates the statements into machine code. In this phase, SAS identifies the type and length of each new variable, and determines whether a variable type conversion is necessary for each subsequent reference to a variable. During the compile phase, SAS creates the following three items:

1. Input buffer :

Is a logical area in memory into which SAS reads each record of raw data when SAS executes an INPUT statement. Note that this buffer is created only when the DATA step reads raw data. (When the DATA step reads a SAS data set, SAS reads the data directly into the program data vector.)

2. Program Data Vector (PDV) :

It is a logical area and virtual memory, which is used for manipulation purpose. It creates two automatically temporary variables.

1. **_N_** : It represent number of iteration done by data steps. Where one iteration equal to one observation.
2. **_ERROR_** : It represent data error in the record has two values.

a) **0** : It represent no error in particular record.

b) **1** : It represent error in the specific record. It does not represent that how many error are there in the record.

3. Descriptor Information :

Is information that SAS creates and maintains about each SAS data set, including data set attributes and variable attributes. It contains, for example, the name of the data set and its member type, the date and time that the data set was created, and the number, names and data types (character or numeric) of the variables.

Descriptor Portion :

- How to see descriptor portion of the dataset in sas ??

```
proc contents data=sashelp.class;
run;
```

The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	02/12/2014 00:07:34	Observation Length	48
Last Modified	02/12/2014 00:07:34	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	19
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASCFG\class.sas7bdat
Release Created	9.0401M1
Host Created	X64_8PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	Age	Num	8
5	Height	Num	8
2	Name	Char	8
1	Obs	Num	8
3	Sex	Char	8
6	Weight	Num	8

Execution Phase:

- The DATA step begins with a DATA statement. Each time the DATA statement executes, a new iteration of the DATA step begins, and the _N_ automatic variable is incremented by 1.
- SAS sets the newly created program variables to missing in the program data vector (PDV).
- SAS reads a data record from a raw data file into the input buffer, or it reads an observation from a SAS data set directly into the program data vector. You can use an INPUT, MERGE, SET, MODIFY, or UPDATE statement to read a record.
- SAS executes any subsequent programming statements for the current record.
- At the end of the statements, an output, return, and reset occur automatically. SAS writes an observation to the SAS data set, the system automatically returns to the top of the DATA step, and the values of variables created by INPUT and assignment statements are reset to missing in the program data vector. Note that variables that you read with a SET, MERGE, MODIFY, or UPDATE statement are not reset to missing here.
- SAS counts another iteration, reads the next record or observation, and executes the subsequent programming statements for the current observation.
- The DATA step terminates when SAS encounters the end-of-file in a SAS data set or a raw data file.

1. Read data step and initializes with missing values.
2. Set statement read one observation
3. Implicit output/Return

PDV (Program Data Vector) :

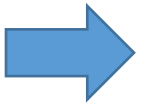
It is a logical area and virtual memory, which is used for manipulation purpose. It creates two automatically temporary variables.

1. **_N_** : It represent number of iteration done by data steps. Where one iteration equal to one observation.
2. **_ERROR_** : It represent data error in the record has two values.

a) **0** : It represent no error in particular record.

b) **1** : It represent error in the specific record. It does not represent that how many error are there in the record.

```
data a;
Put _all_;
set sashelp.class;
put _all_;
run;
```



```
Obs=. Name=  Sex=  Age=. Height=. Weight=. _ERROR_=0 _N_=1
Obs=1 Name=Alfred Sex=M Age=14 Height=69 Weight=112.5 _ERROR_=0 _N_=1
Obs=1 Name=Alfred Sex=M Age=14 Height=69 Weight=112.5 _ERROR_=0 _N_=2
Obs=2 Name=Alice Sex=F Age=13 Height=56.5 Weight=84 _ERROR_=0 _N_=2
Obs=2 Name=Alice Sex=F Age=13 Height=56.5 Weight=84 _ERROR_=0 _N_=3
```

PDV (Program Data Vector) :

Name	Age	Gender
Ram	21	M
Sita	20	F
Radha	19	F

Program:-

A=Data one**;**

B=Set test**;**

C=run;

➢1. Read data step and initialize with missing value.

	Name	Age	Gender	_Error_	_N_
A		.		0	1

➢2.Set statement read one observation

	Name	**Age**	**Gender**	**Error**	**_N_**
A		.		0	1
B	**Ram**	**21**	**M**	**0**	**1**

➢3. Implicit output

	Name	**Age**	**Gender**
C	Ram	21	M

➢4. Implicit Return

	Name	**Age**	**Gender**	**Error**	**_N_**
A		.		0	1
B	Ram	21	M	0	1
A	Ram	21	M	0	2

IF/Where/Keep/Drop Statement:

WHERE/(Keep/Drop in set statement)

Keep/Drop in dataset Statement

IF/Keep/Drop Statements

IF/Where/Keep/Drop Statement:

```
Data one;
set sashelp.class;
run;

Data two;
set sashelp.class;
Keep Name age ;
run;

Data three;
set sashelp.class(keep=Name age);
run;

Data four(keep=Name age);
set sashelp.class;
run;
```

```
Data four1(drop=sex);
set sashelp.class(Drop=Height Weight);
Salary= age*1000;
drop age;
run;

Data five;
set sashelp.class;
if age >14;
run;

Data six;
Set sashelp.class;
where age > 14;
run;
```

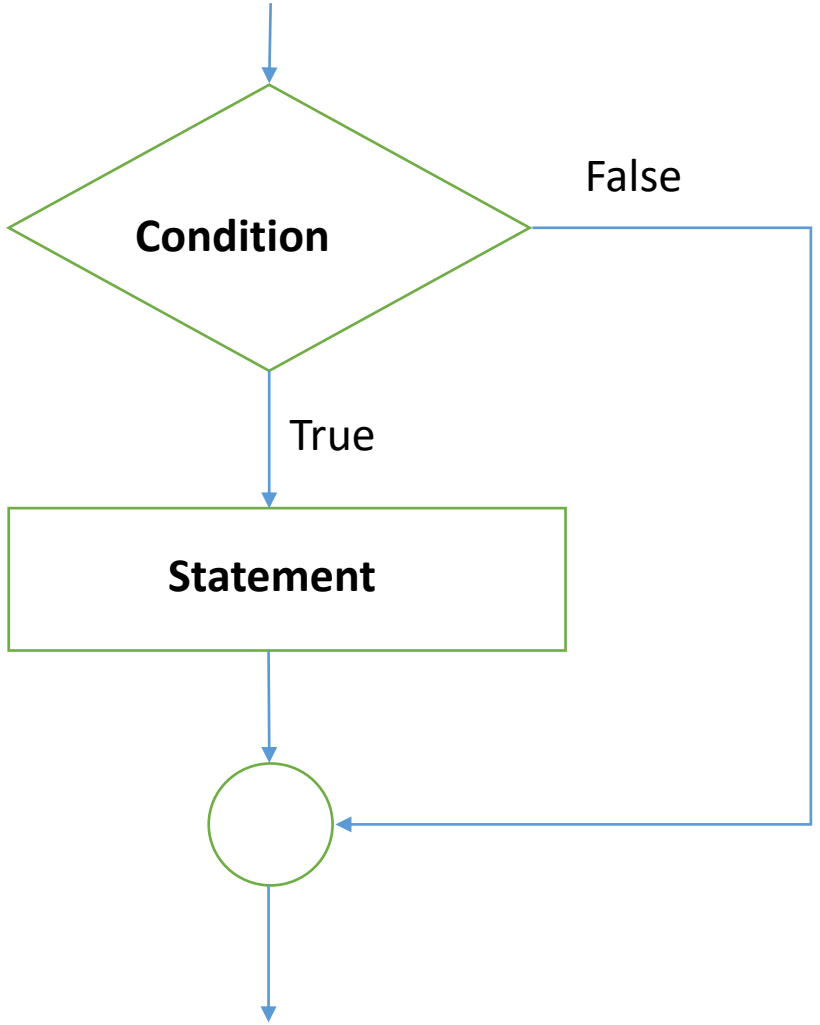
IF -THEN / IF -THEN –Else Statement :

IF Condition/expression THEN statement;

IF Condition/expression THEN statement1;
Else IF Condition/expression THEN statement2;
Else Statement3;

```
Data six1;
Set sashelp.class;
if age=15 then salary=age*1000;
run;
```

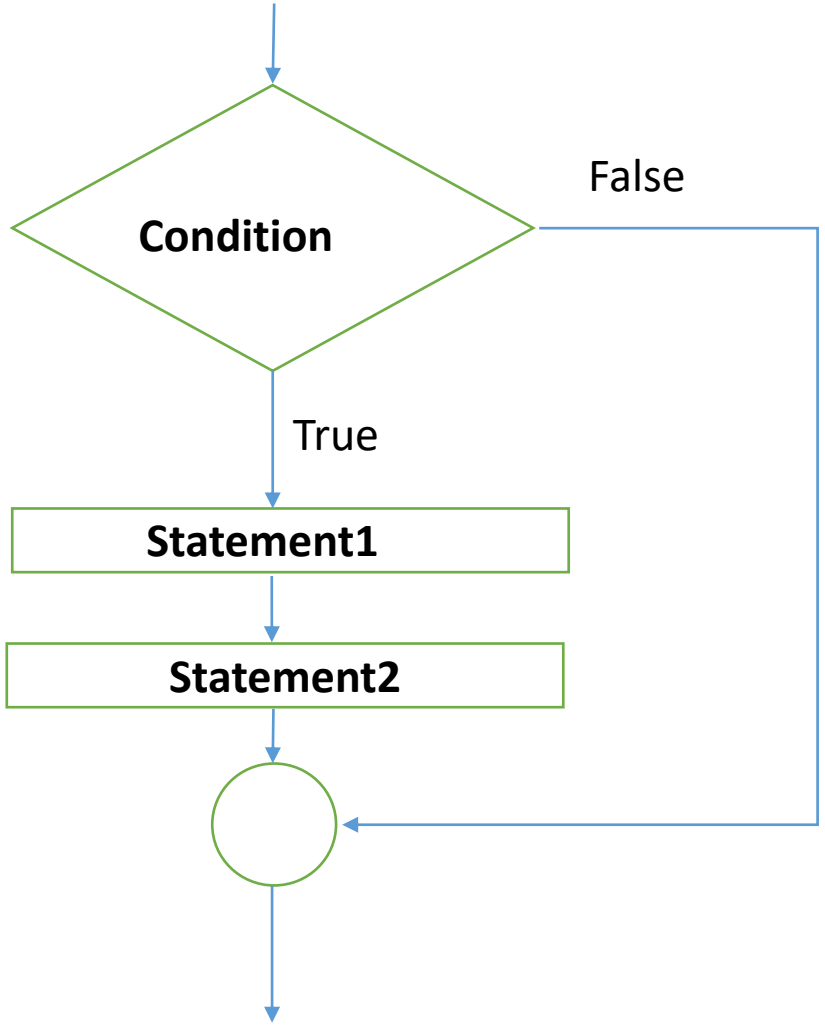
```
Data six1;
Set sashelp.class;
if age < 15 then salary=age*1000;
else if age=15 then salary=age*2000;
else salary=age*3000;
run;
```



IF -THEN-DO Statement :

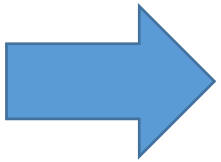
```
IF Condition/expression THEN do;
        statement1;
        statement2;
End;
-----------------------------------------
IF Condition/expression THEN do;
        statement1;
        statement2;
End;
Else IF Condition/expression THEN do;
        statement1;
        statement2;
End;
```

```
Data six11;
Set sashelp.class;
if age=15 then do;
 Salary=age*1000;
Bonus= age *10;
End;
run;
```



Sort / Order the dataset

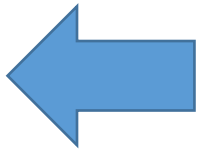
```
data a;
set sashelp.class;
run;


proc sort data=a out=b;
by age;
run;
```



Obs	Name	Sex	Age
1	Alfred	M	14
2	Alice	F	13
3	Barbara	F	13
4	Carol	F	14
5	Henry	M	14
6	James	M	12
7	Jane	F	12
8	Janet	F	15
9	Jeffrey	M	13
10	John	M	12
11	Joyce	F	11
12	Judy	F	14
13	Louise	F	12
14	Mary	F	15
15	Philip	M	16
16	Robert	M	12
17	Ronald	M	15
18	Thomas	M	11
19	William	M	15

	Obs	Name	Sex	Age	Height
1	11	Joyce	F	11	
2	18	Thomas	M	11	
3	6	James	M	12	
4	7	Jane	F	12	
5	10	John	M	12	
6	13	Louise	F	12	
7	16	Robert	M	12	
8	2	Alice	F	13	
9	3	Barbara	F	13	
10	9	Jeffrey	M	13	
11	1	Alfred	M	14	
12	4	Carol	F	14	
13	5	Henry	M	14	
14	12	Judy	F	14	
15	8	Janet	F	15	
16	14	Mary	F	15	
17	17	Ronald	M	15	
18	19	William	M	15	
19	15	Philip	M	16	

If can also alter the default ascending order by using descending followed by var name.



Age
11
11
12
12
12
12
12
13
13
13
14
14
14
14
15
15
15
15
16

	Age
1	16
2	15
3	15
4	15
5	15
6	14
7	14
8	14
9	14
10	13
11	13
12	13
13	12
14	12
15	12
16	12
17	12
18	11
19	11

```
Proc sort data=a(keep= age) out=c(keep=age);
by descending age;
run;
```

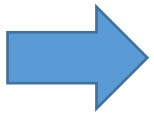
Caution: If you don't specify out= then parent dataset will be changed

Presented By : Shashi Kumar
YouTube Channel : https://lnkd.in/fNSUTDE

nodupkey, nodup & nouniquekey options

Three

	eid	name	sex
1	1	A	M
2	1	A	F
3	2	B	M
4	2	C	M
5	3	D	F
6	3	D	M
7	1	E	M
8	1	A	M
9	2	F	M
10	2	C	M
11	4	G	M
12	5	H	M
13	5	H	F



Threes

	eid	name	sex
1	1	A	M
2	1	A	F
3	1	E	M
4	1	A	M
5	2	B	M
6	2	C	M
7	2	F	M
8	2	C	M
9	3	D	F
10	3	D	M
11	4	G	M
12	5	H	M
13	5	H	F

Four

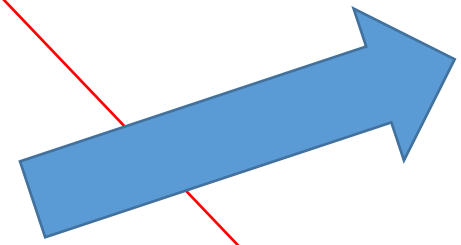
	eid	name	sex
1	1	A	M
2	2	B	M
3	3	D	F
4	4	G	M
5	5	H	M

proc sort data=three out=four nodupkey;
by eid;
run;

proc sort data=three out=threes;
by eid;**run**;

nodupkey= it keep the first unique observation according to key variable(variable present in the by statement)

	eid	name	sex
1	1	A	M
2	1	A	F
3	1	E	M
4	1	A	M
5	2	B	M
6	2	C	M
7	2	F	M
8	2	C	M
9	3	D	F
10	3	D	M
11	4	G	M
12	5	H	M
13	5	H	F



FIVE

	eid	name	sex
1	1	A	F
2	1	A	M
3	1	E	M
4	2	B	M
5	2	C	M
6	2	F	M
7	3	D	F
8	3	D	M
9	4	G	M
10	5	H	F
11	5	H	M

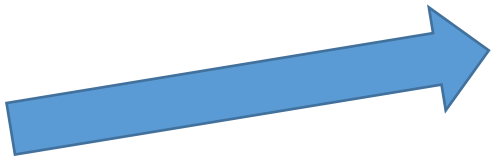
	eid	name	sex
1	1	A	M
2	2	C	M

SIX

```
proc sort data=three out=five nodup dupout=six;
by _all_;run;
```

nodup=It keep the first unique observation, if entire row is duplicate

	eid	name	sex
1	1	A	M
2	1	A	F
3	1	E	M
4	1	A	M
5	2	B	M
6	2	C	M
7	2	F	M
8	2	C	M
9	3	D	F
10	3	D	M
11	4	G	M
12	5	H	M
13	5	H	F



Seven

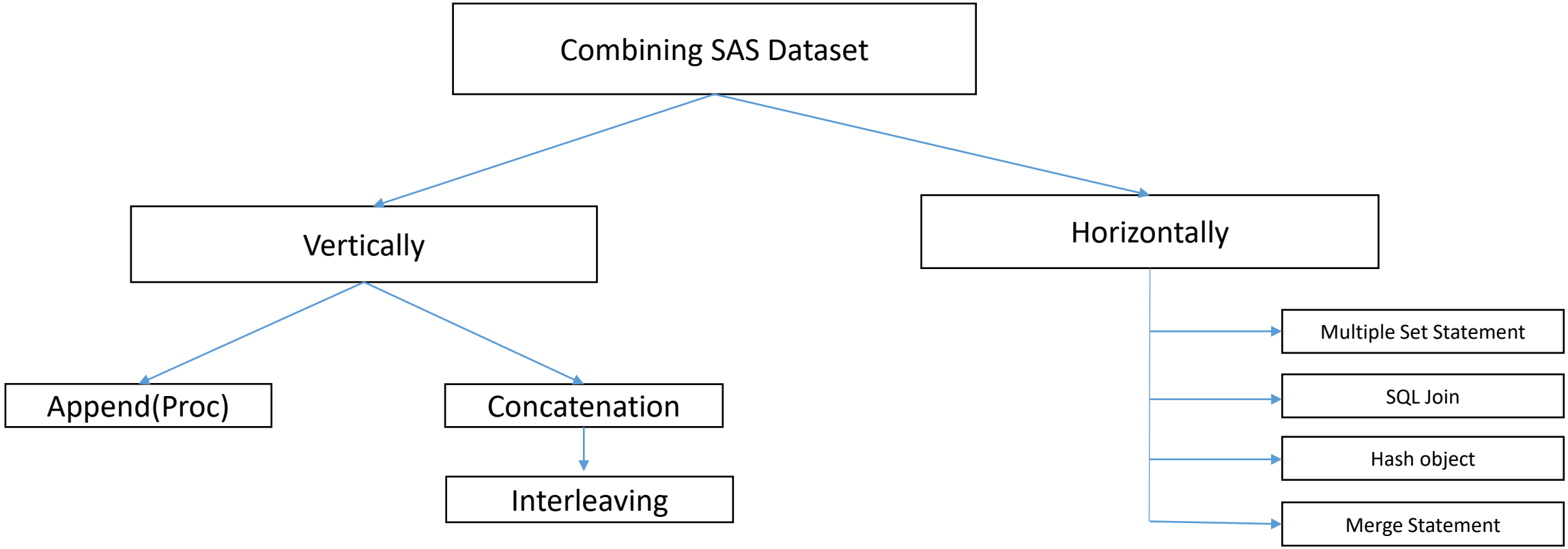
	eid	name	sex
1	1	A	M
2	1	A	F
3	1	E	M
4	1	A	M
5	2	B	M
6	2	C	M
7	2	F	M
8	2	C	M
9	3	D	F
10	3	D	M
11	5	H	M
12	5	H	F

	eid	name	sex
1	4	G	M

Eight

proc sort data=three out=seven nouniquekey uniqueout=eight;
by eid;**run**;

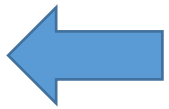
- nouniquekey = It eliminate unique records based on by variable.
- Duplicate record deleted from original dataset can be saved in another dataset by using the option uniqueout.


Combining SAS Dataset
Vertically
Horizontally
Append(Proc)
Concatenation
Interleaving
Multiple Set Statement
SQL Join
Hash object
Merge Statement

Appending & Interleaving

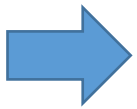
- Appending is joining two datasets vertically.
- Mention dataset name/s in single set statement to append.
- Make sure the variable/s names should be same otherwise unwanted result would come.
- Interleaving combines individual sorted dataset into one sorted dataset by specified variable in the by statement.

card	amount
1	100
1	200
1	100
2	200
2	300
1	100
1	200
1	100
2	200
2	300



```
data append;
set a a;
run;
```

```
data interleave;
set a a;
by card;
run;
```



card	amount
1	100
1	200
1	100
1	100
1	200
1	100
2	200
2	300
2	200
2	300

Merging

What's need to have a successful merging??

- There should be at least one key variable else unwanted result will populate.
- Key variable values should be sorted in both dataset.
- Key variable attribute i.e format, length , alignment should be same.

- ❑zero-to-one
- ❑one-to-zero
- ❑ one-to-one
- ❑one-to-many
- ❑ many-to-one
- ❑ few-to-many
- ❑many-to-few
- ❑many-to-many

Types of Matched Merging

Merging

Different Cases of Merging

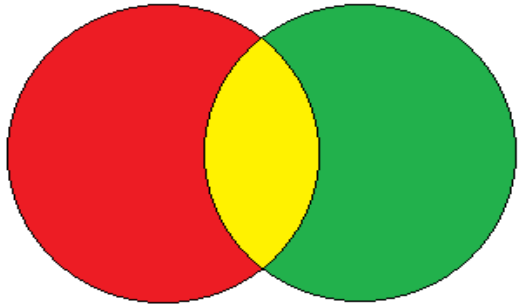
- You can do it by merge statement
- You can do it also with set statement
- Be very careful when you have different # of records in both tables i.e in few to many or few to many cases

Data one		Data Two		Merging Type
ID	Var1	ID	Var2	
A01	53	A01	40	Zero to One
A02	24	A02	62	
A05	86	A04	71	One to Zero
A10	64	A10	40	
A25	18	A25	27	One to One
A25	96	A25	72	
A25	66	A25	90	One to Many
A25	41	A25	58	
A32	63	A32	10	Many to One
A55	16	A32	20	
A55	51	A32	19	Few to Many
A55	61	A55	99	
A92	40	A92	71	Many to Few
A92	34	A92	87	
A96	11	A92	51	Mant to Many
A96	56	A96	94	
A96	50	A96	31	

Full Join

```
data M;
merge A (in = x) B (in = y);
by id;
if x = 1 or y = 1;
run;
```

```
Proc sql;
create table M2 as
select coalesce(a.id, b.id) as id, gender, sex
from A full join B on a.id = b.id;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

	ID	Gender	Sex
1	1	M	
2	2	M	
3	3	M	
4	4	M	
5	5	M	
6	6	F	F
7	7	F	F
8	8	M	M
9	9	M	M
10	10	M	M
11	11		F
12	12		F
13	13		F
14	14		F
15	15		F

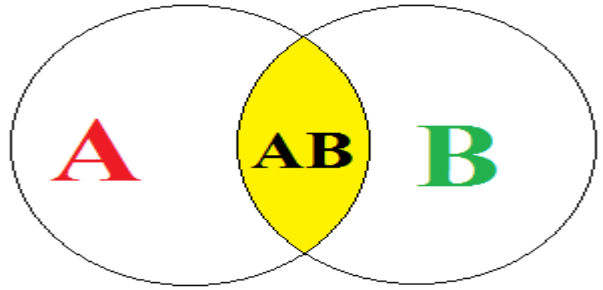
A

B

Inner Join

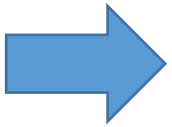
```
data M;
merge A (in = x) B (in = y);
by id;
if x = 1 and y = 1;
run;
```

```
Proc sql;
create table M2 as
select coalesce(a.id, b.id) as id, gender, sex
from A inner join B on a.id = b.id;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

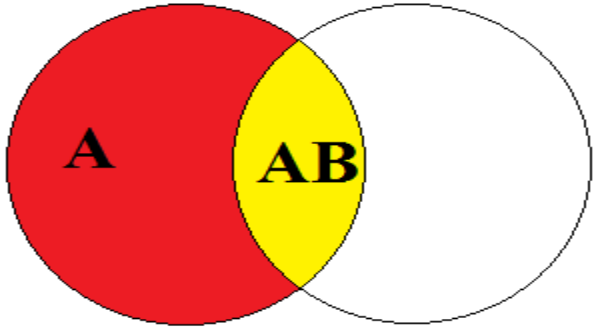


	ID	Gender	Sex
1	6	F	F
2	7	F	F
3	8	M	M
4	9	M	M
5	10	M	M

Left Join

```
data M;
merge A (in = x) B (in = y);
by id;
if x = 1 ;
run;
```

```
Proc sql;
create table M2 as
select coalesce(a.id, b.id) as id, gender, sex
from A left join B on a.id = b.id;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

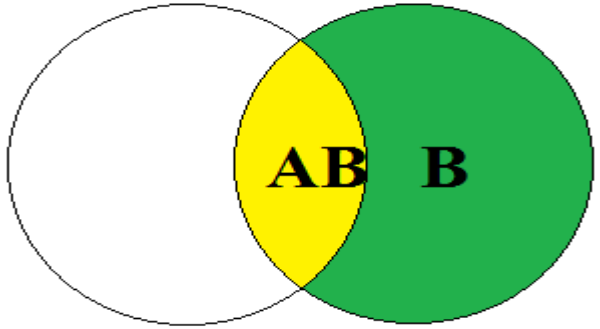
	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

	ID	Gender	Sex
1	1	M	
2	2	M	
3	3	M	
4	4	M	
5	5	M	
6	6	F	F
7	7	F	F
8	8	M	M
9	9	M	M
10	10	M	M

Right Join

```
data M;
merge A (in = x) B (in = y);
by id;
if y = 1 ;
run;
```

```
Proc sql;
create table M2 as
select coalesce(a.id, b.id) as id, gender, sex
from A right join B on a.id = b.id;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

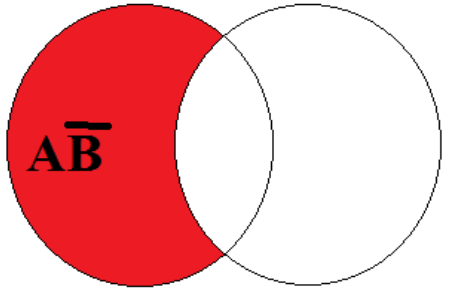
	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

	ID	Gender	Sex
1	6	F	F
2	7	F	F
3	8	M	M
4	9	M	M
5	10	M	M
6	11		F
7	12		F
8	13		F
9	14		F
10	15		F

Non Matching From A

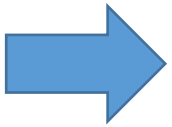
```
data M;
merge A (in = x) B (in = y);
by id;
If  x=1 and y = 0 ;
run;
```

```
Proc Sql;
create table Q2 as
select coalesce(a.id, b.id) as id, gender, sex
from A left join B on a.id = b.id
where b.id is null;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

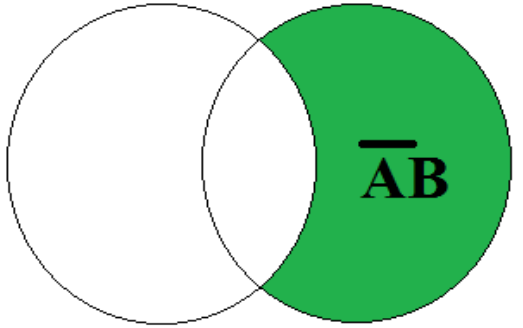


	ID	Gender	Sex
1	1	M	
2	2	M	
3	3	M	
4	4	M	
5	5	M	

Non Matching From B

```
data M;
merge A (in = x) B (in = y);
by id;
If  x=0 and y = 1 ;
run;
```

```
Proc Sql;
create table Q2 as
select coalesce(a.id, b.id) as id, gender, sex
from A right join B on a.id = b.id
where a.id is null;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

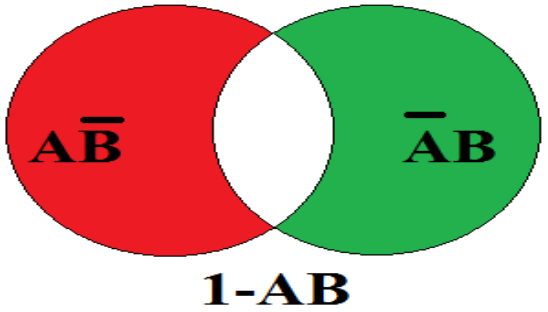
	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

	ID	Gender	Sex
1	11		F
2	12		F
3	13		F
4	14		F
5	15		F

Non Matching From A and B

```
data M;
merge A (in = x) B (in = y);
by id;
If  (x=0 and y = 1) or (y=0 and x = 1) ;
run;
```

```
Proc Sql;
create table Q2 as
select coalesce(a.id, b.id) as id, gender, sex
from A right join B on a.id = b.id
where a.id is null or b.id is null ;
quit;
```



	ID	Gender
1	1	M
2	2	M
3	3	M
4	4	M
5	5	M
6	6	F
7	7	F
8	8	M
9	9	M
10	10	M

	ID	Sex
1	6	F
2	7	F
3	8	M
4	9	M
5	10	M
6	11	F
7	12	F
8	13	F
9	14	F
10	15	F

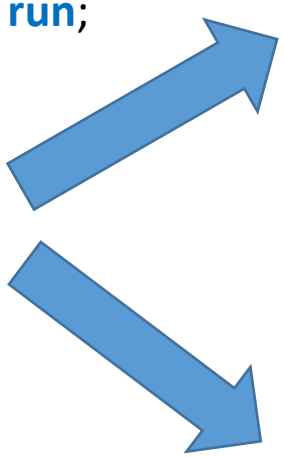
	ID	Gender	Sex
1	1	M	
2	2	M	
3	3	M	
4	4	M	
5	5	M	
6	11		F
7	12		F
8	13		F
9	14		F
10	15		F

Many to Many

	ID	Info
1	1	3123
2	1	1234
3	2	7482
4	2	8912
5	3	1284

	ID	Info2
1	1	4444
2	1	5555
3	1	8989
4	2	9099
5	2	8888
6	3	8989

```
data combined;
merge dat1 dat2 ;
by ID;
run;
```



	ID	Info	Info2
1	1	3123	4444
2	1	1234	5555
3	1	1234	8989
4	2	7482	9099
5	2	8912	8888
6	3	1284	8989

	ID	Info	Info2
1	1	3123	4444
2	1	1234	4444
3	1	3123	5555
4	1	1234	5555
5	1	3123	8989
6	1	1234	8989
7	2	7482	9099
8	2	8912	9099
9	2	7482	8888
10	2	8912	8888
11	3	1284	8989

```
proc sql ;
create table combined2 as
select coalesce(dat1.id, dat2.id) as id ,info,info2
from dat1  full join dat2  on dat1.ID = dat2.ID;
quit;
```

Working with Date/Time/Date Time :

SAS Date value
is a value that represents the number of days between January 1, 1960, and a specified date. SAS can perform calculations on dates ranging from A.D. 1582 to A.D. 19,900. Dates before January 1, 1960, are negative numbers; dates after January 1, 1960, are positive numbers.

SAS time value
is a value representing the number of seconds since midnight of the current day. SAS time values are between 0 and 86400

SAS datetime value
is a value representing the number of seconds between January 1, 1960, and an hour/minute/second within a specified date.

```
data a;
format y datetime22.;
do i= 0 to 10;
y=i;
output;
end;
run;
```

	y	i
1	01JAN1960:00:00:00	0
2	01JAN1960:00:00:01	1
3	01JAN1960:00:00:02	2
4	01JAN1960:00:00:03	3
5	01JAN1960:00:00:04	4
6	01JAN1960:00:00:05	5

```
data a;
format y date9.;
do i= -1 to 5;
y=i;
output;
end;
run;
```

31DEC1959	-1
01JAN1960	0
02JAN1960	1
03JAN1960	2
04JAN1960	3
05JAN1960	4
06JAN1960	5

```
data a;
format y time.;
do i= 0 to 10;
y=i;
output;
end;
run;
```

	y	i
1	0:00:00	0
2	0:00:01	1
3	0:00:02	2
4	0:00:03	3
5	0:00:04	4
6	0:00:05	5

Formats Date/Time/Date Time :

Format : Which is used to convert standard data to non standard data.
Informant : Which is used to convert Non Standard data into standard data.

10/09/08 : Non-Standard :::: Informant ::::::MMDDYY8.
$40,000 : Non-Standard :::: Informant ::::::Dollar7.
74,000 : Non-Standard :::: Informant ::::::Comma6.
34555 : Standard :::::::::: Format :::::::: ?? (Comma6., MMDDYY8.,date9……)

Format variable name <$> formatW.d;
W : Total Width
d : Number of decimal place
$: It indicate Character format

i.e, Format Doj **date9.**;
(Format either SAS built or User define format (Proc format))

```
*Convert Non-standard DOJ to Standard DOJ;
data one;
infile datalines dlm="";
input Name$ Doj;
informat Doj DDMMYY8.;
datalines;
Shashi 10/09/19
Ravi 21/12/19
;
run;
```

	Name	Doj
1	Shashi	21802
2	Ravi	21904

```
*Convert Non-standard DOJ to Standard DOJ
  and standard DOJ to Non-Standard DOJ;
data two;
infile datalines dlm="";
input Name$ Doj;
informat Doj DDMMYY8.;
format Doj date9.;
datalines;
Shashi 10/09/19
Ravi 21/12/19
;
run;
```

	Name	Doj
1	Shashi	10SEP2019
2	Ravi	21DEC2019

Formats Date/Time/Date Time :

Input	Format		Output
1	Date7.	1	01JAN60
1	Date9.	1	01JAN1960
1	DDMMYY10.	1	01/01/1960
1	DDMMYY.	18703	17/03/11
1	DDMMYY10.	18703	17/03/2011
1	DDMMYYB.	18703	17 03 11
1	DDMMYYB10.	18703	17 03 2011

There are various date / Time / Date time formats
as per your need your choose your format.
Especially in transections data datetime format is datetime22.

Format	Input	Output
HHMM.	53132	14:46
HOUR.	53132	15
MMSS.	53132	885
TIME.	53132	14:45:32
TOD.	53132	14:45:32

Formats Date/Time/Date Time :

1. **Today** :- Return a current date from a SAS date value.
2. **Day** :- Extract the day of the month from a SAS date and returns a number from 1-31.
3. **Weekdays** :- Returns the day of the week from SAS date and return a number from 1 to 7.
   (Sunday =1; Monday=2;Tuesday=3;Wednesday=4;Thrusday=5;Friday=6;Saturday=7)
4. **Month** :- Extract the month from the SAS date and return a number from 1 to 12.
   (January =1;February=2;March=3;April=4……December=12)
5. **Year** :- Extract the year from the SAS date and returns 4 digit of years.
6. **Qtr** :- Extract the quarter from the SAS date and returns number from 1-4.
   (Jan- March = 1;Apr-June=2;July-Sep=3;October-December=4)
7. **MDY** :- Return a SAS date value from numeric month, day and year value.

Function	Input	Output
Today	Current System Date	21534
Day	13FEB2019	13
Weekday	13FEB2019	4
Month	13FEB2019	2
Qtr	13FEB2019	1
Year	13FEB2019	2019
MDY	(02,13,2019) *Arg should be numeric	21534 (Days From 1 Jan 1960)

Formats Date/Time/Date Time :

	DATE	international airline travel (thousands)	A	B	C	D	E	F	G	H	I	J
1	JAN49	112	22DEC2018	1	7	1	1	1949	01JAN1960	01JAN1949	0	Saturday, Jan 1, 1949
2	FEB49	118	22DEC2018	1	3	2	1	1949	01JAN1960	01FEB1949	31	Tuesday, Feb 1, 1949
3	MAR49	132	22DEC2018	1	3	3	1	1949	01JAN1960	01MAR1949	60	Tuesday, Mar 1, 1949
4	APR49	129	22DEC2018	1	6	4	2	1949	01JAN1960	01APR1949	91	Friday, Apr 1, 1949
5	MAY49	121	22DEC2018	1	1	5	2	1949	01JAN1960	01MAY1949	121	Sunday, May 1, 1949
6	JUN49	135	22DEC2018	1	4	6	2	1949	01JAN1960	01JUN1949	152	Wednesday, Jun 1, 1949
7	JUL49	148	22DEC2018	1	6	7	3	1949	01JAN1960	01JUL1949	182	Friday, Jul 1, 1949
8	AUG49	148	22DEC2018	1	2	8	3	1949	01JAN1960	01AUG1949	213	Monday, Aug 1, 1949
9	SEP49	136	22DEC2018	1	5	9	3	1949	01JAN1960	01SEP1949	244	Thursday, Sep 1, 1949
10	OCT49	119	22DEC2018	1	7	10	4	1949	01JAN1960	01OCT1949	274	Saturday, Oct 1, 1949
11	NOV49	104	22DEC2018	1	3	11	4	1949	01JAN1960	01NOV1949	305	Tuesday, Nov 1, 1949
12	DEC49	118	22DEC2018	1	5	12	4	1949	01JAN1960	01DEC1949	335	Thursday, Dec 1, 1949
13	JAN50	115	22DEC2018	1	1	1	1	1950	01JAN1960	01JAN1950	0	Sunday, Jan 1, 1950
14	FEB50	126	22DEC2018	1	4	2	1	1950	01JAN1960	01FEB1950	31	Wednesday, Feb 1, 1950
15	MAR50	141	22DEC2018	1	4	3	1	1950	01JAN1960	01MAR1950	60	Wednesday, Mar 1, 1950
16	APR50	135	22DEC2018	1	7	4	2	1950	01JAN1960	01APR1950	91	Saturday, Apr 1, 1950
17	MAY50	125	22DEC2018	1	2	5	2	1950	01JAN1960	01MAY1950	121	Monday, May 1, 1950
18	JUN50	149	22DEC2018	1	5	6	2	1950	01JAN1960	01JUN1950	152	Thursday, Jun 1, 1950
19	JUL50	170	22DEC2018	1	7	7	3	1950	01JAN1960	01JUL1950	182	Saturday, Jul 1, 1950

```
data test;
set sashelp.air;
A=today();
B=day(date);
C=weekday(date);
D=month(date);
E=qtr(date);
F=year(date);
G=MDY(1,1,1960);
H=MDY(D,B,F);
I=MDY(month(date),1,year('08JAN1960'd));
J=date;
format A G H date9. J weekdate24.;
run;
```

Character and Numeric Functions:

Trim(variable) *Remove trailing Blank	Cmiss(variable/vector) *count of Missing across row both char & num. Variable.
Strip(variable) *Remove Leading and Trailing Blank	n() * count non missing values
Left(variable) * Left Align a Character String	Scan(String,nth word,delimeter) * Return nth word of the character value
Right(variable) * Right Align a Character String	Find(string,substring,modifier,start position)* It search a target string to specified substring and return numeric value.
Lowcase(variable) * Convert in Low Case	Cat(String1,String2,…Stringn) *Doesnot renove leading and trailing blank before concatenate
Upcase(variable) * Convert in Up Case	Catt(String1,String2,…Stringn) * Remove trailing blank before concatenate
Propcase(variable) * Convert in 1st character in up case and reaming low case	Cats(String1,String2,…Stringn) * Remove leading blank before concatenate
Length(variable) * Return total number of column width	Catx(delimeter,String1,String2,…Stringn) * Remove leading and trailing blank and add delimiter between string
Char(variable,pos) *Return a single character from specified position in a character String	Tranwrd (Source,target,replacement) *Search and replace from character string
Int(variable)	Input(Source,Informat) * Convert to numeric
Ceil(variable)	Put(Source,format) * Round and convert to character
Floor(variable)	Substr(String,Start position,Length) * Extract character from string
Round(variable)	Compress(Source,Character,modifier) *Removes the characters listed in the character argument from the source.
Nmiss(variable/vector) *count of Missing across row both numeric variable	Compbl(String) * Remove multiple blank from a String by translating each occurrence of two or more conjugative blank into single blank.

Character and Numeric Functions:

```
data test;
x=' Ram ';
y=' Sita ';
A=trim(x);
B=strip(x);
C=left(x);
D=right(x);
E=upcase(x);
F=lowcase(x);
g=length(x);
H=x!!y;
I=length(H);
J=char(x,2);
K=length(x);
L=length(D);
run;
```

	x	y	A	B	C	D	E	F	g	H	I	J	K	L
1	Ram	Sita	Ram	Ram	Ram	Ram	RAM	ram	4	Ram Sita	10	R	4	5

proc contents data=test; run;

Variable	Type	Len
A	Char	5
B	Char	5
C	Char	5
D	Char	5
E	Char	5
F	Char	5
H	Char	11
I	Num	8
J	Char	1
K	Num	8
L	Num	8
g	Num	8
x	Char	5
y	Char	6

Character and Numeric Functions:

```
data test2;
infile datalines;
input num;
A=int(num);
B=ceil(num);
C=Floor(num);
D=round(num);
E=round(num,5);
F=round(num,11);
G=round(num,.33);
datalines;
10
15
32.5
79
37.9
-10
-23.6
;
run;
```

num	A	B	C	D	E	F	G
10	10	10	10	10	10	11	9.9
15	15	15	15	15	15	11	14.85
32.5	32	33	32	33	35	33	32.34
79	79	79	79	79	80	77	78.87
37.9	37	38	37	38	40	33	37.95
-10	-10	-10	-10	-10	-10	-11	-9.9
-23.6	-23	-23	-24	-24	-25	-22	-23.76

If A is +ve then int(num)=floor(num)
If A is -ve then int(num)=Ceil(num)
Round: convert to nearest integer with multiple of 2nd argument.

Variable	Type	Len
A	Num	8
B	Num	8
C	Num	8
D	Num	8
E	Num	8
F	Num	8
G	Num	8
num	Num	8

Character and Numeric Functions:

```
data test3;
name='Shashi Kumar';
x=substr(name,1,2);
y=substr(name,3,2);
z=substr(name,1,7);

A=substr('Mohan',3,1);
B=substr(upcase(name),3,4);
run;
```

name	x	y	z	A	B
Shashi Kumar	Sh	as	Shashi	h	ASHI

Variable	Type	Len
A	Char	5
B	Char	12
name	Char	12
x	Char	12
y	Char	12
z	Char	12

Character and Numeric Functions:

```
data test4;
A='Today is FRIDAY';
B=scan(A,2);
C=scan(A,1,'a');
D=scan(A,-1);
E=find(A,'a');
F=find(A,'A');
G=find(A,'a',7);
H=find(A,'a','i',7);
I=' Ram ';
J=' Sita ';
K=cat(I,J,I);
L=catt(I,J,I);
M=cats(I,J,I);
N=catx('/',I,J,I);
O=catx('0',I,J,I);
P=tranwrd(N,'/','*');
Q=tranwrd(N,'Ram','Sita');
R='$500';
S=0;
T=put(S,date9.);
U=input(R,dollar4.);
run;
```

A	B	C	D
Today is FRIDAY	is	Tod	FRIDAY

E	F	G	H
4	14	0	14

I	J	K	L	M	N	O
Ram	Sita	Ram Sita Ram	Ram Sita Ram	RamSitaRam	Ram/Sita/Ram	Ram0Sita0Ram

P	Q	R	S	T	U
Ram*Sita*Ram	Sita/Sita/Sita	$500	0	01JAN1960	500

Variable	Type	Len
A	Char	15
B	Char	200
C	Char	200
D	Char	200
E	Num	8
F	Num	8
G	Num	8
H	Num	8
I	Char	5
J	Char	6
K	Char	200
L	Char	200
M	Char	200
N	Char	200
O	Char	200
P	Char	200
Q	Char	200
R	Char	4
S	Num	8
T	Char	9
U	Num	8

Character and Numeric Functions:

*%%%%%%%%%%%%%%%%%%%% compress %%%%%%%%%%%%%;

data test8;

string='**StudySAS Blog! 17752 ** ';
string1=compress(string,'') ; *Compress spaces. This is default;
string2=compress(string,'','ak'); *Compress alphabetic chars(1,2etc);
string3=compress(string,'','d') ; *Compress numerical values;
string4=compress(string,'','l'); *Compress lowercase characters;
string5=compress(string,'','u'); *Compress uppercase characters;
string6=compress(string,'S','k'); *Keeps only specified characters;
string7=compress(string,'!.','P'); *Compress Punctuations only;
string8=compress(string,'s','i'); *upper/lower case specified characters;
string9=compress(string,'','a'); *Compress all upper\lower case characters ;
string10=compress(string,'','s') ; * Compress or delete spaces;
string11=compress(string,'','kd') ; *Compress alphabets (Keeps only digits);

run ;

string	string1	string2	string3	string4
StudySAS Blog! 17752.	StudySASBlog!17752.	StudySAS Blog	StudySASBlog!.	SSASB!17752.

string5	string6	string7	string8
tudylog!17752.	SSS	StudySAS Blog 17752	tudyA Blog! 17752.

string9	string10	string11
!17752.	StudySASBlog!17752.	17752

Descriptive Statistics : Proc Freq

Proc freq data= input data <option>;
Table var1 var2 …..varn;
Run;

1. The freq procedure produces one way to n-way frequency tables.
2. The table statement specifies the frequency table and cross tabulation to produce.
3. * between variable request n-way cross tabulation tables.
4. One way frequency tables produces ***freq***, ***cumulative freq***, ***percentage***,***Cummulative percentage***.
5. N-way freq table produces ***frequency,row%,column %,Total %***

Note:- Without the table statement Proc freq produces the frequency table for each variable (Character & Numeric).

Option can be Placed in table statement after a / to suppressed the display of default statistics.

1. Nocum
2. NoPercent
3. NoFreq
4. Norow
5. Nocol

Options to be added in table statement after the / to control the dataset.

1. Outcome : Include the ***cumulative freq*** and ***cumulative percentage*** in output data set.
2. Outpct : Include the ***column%*** and ***row %*** in the output dataset.

Descriptive Statistics : Proc means

Proc Means data = input data <option>;
Var analysis variable;
Class Classification Variable;
Run;

It provides data summarization tools to compute descriptive statistics for variables across all observation and with group of observation.

1. The means procedure produces summary report that display descriptive statistics.
2. The var statement specifies the analysis variable and their order in the result.
3. The class statement identifies the variables whose value is defined subgroups for the analysis.
4. By default the means procedure create the report with ***N, mean, Standard deviation, Minimum, Maximum***.

Note: Without the Var statement proc means analysis all numeric variables in the data set.

Length/Label/Attrib Statement

1. Length var<$> length;

length name $ 5;
Length age 3;

1. Length statement defines length of the variables.
2. Length of character variable must be define before the variable created at PDV.

2. Label var1= “label 1” var2= “label 2” … varn= “label n” ;

1. It assign the descriptive level to the variable name.
2. Any number of variables can be associated with single label statement;
3. A label can have **256** character;
4. Using a label statement in the data step, Permanently associate labels with variable by storing the label in the description portion of SAS data set.

Proc print Option:-

1. **Label** : By default proc print the variable name in the output window. If we need to print the label then we have to used label option in proc print statement.

2. **Split**: It is used to split the label in to multiple line.

Presented By : Shashi Kumar

3. Attrib variable-list attribute-list ;

Associates a format, informat, label, and length with one or more variables.

```
data attrib1;
set sashelp.class;
attrib Age  Label="Student Age" length =5  format=dollar5.
       Name Label="Student Name";
run;
```

```
data label1;
 set sashelp.class;
 run;
proc contents data=label1;run;

data label2;
 set sashelp.class;
 label name="Student Name" sex="Gender";
 run;
proc contents data=label2;run;

proc print data=sashelp.class label;
 label name="Student Name";
 run;
proc print data=sashelp.class split='*';
 label name="Student*Name";
 run;
```

First dot and last dot (By grouping Processing) :

1. The By statement in dataset enable SAS to process data in groups.
2. By statement data step create two temporary variable for each variable listed in the by statement.
3. The first variable has a value of 1 for the first observation in the by group, otherwise it equal to 0.
4. The last variable has a value of 1 for the last observation in the by group otherwise it is 0.

Descriptive Statistics : Proc Freq

Proc freq data= input data <option>;
Tables var1 var2 …..varn;
Run;

1. The freq procedure produces one way to n-way frequency tables.
2. The table statement specifies the frequency table and cross tabulation to produce.
3. ***** between variable request n-way cross tabulation tables.
4. One way frequency tables produces ***freq***, ***cumulative freq***, ***percentage***,***Cummulative percentage***.
5. N-way freq table produces ***frequency,row%,column %,Total %***

Note:- Without the table statement Proc freq produces the frequency table for each variable.

Option can be Placed in table statement after a / to suppressed the display of default statistics.

1. Nocum
2. NoPercent
3. NoFreq
4. Norow
5. Nocol

Options to be added in table statement after the / to control the dataset.

1. Outcome : Include the ***cumulative freq*** and ***cumulative percentage*** in output data set.
2. Outpct : Include the ***column%*** and ***row %*** in the output dataset.

Descriptive Statistics : Proc means

Proc Means data = input data <option>;
Var analysis variable;
Class Classification Variable;
Run;

It provides data summarization tools to compute descriptive statistics for variables across all observation and with group of observation.

1. The means procedure produces summary report that display descriptive statistics.
2. The var statement specifies the analysis variable and their order in the result.
3. The class statement identifies the variables whose value is defined subgroups for the analysis.
4. By default the means procedure create the report with ***N, mean, Standard deviation, Minimum, Maximum***.

Note: Without the Var statement proc means analysis all numeric variables in the data set.

Length Statement

Length var<$> length;

i.e ;
length name $ 5;
Length age 3;

1. Length statement defines length of the variables.
2. Length of character variable must be define before the variable created at PDV.

First dot and last dot (By grouping Processing) :

1. The By statement in dataset enable SAS to process data in groups.
2. By statement data step create two temporary variable for each variable listed in the by statement.
3. The first variable has a value of 1 for the first observation in the by group, otherwise it equal to 0.
4. The last variable has a value of 1 for the last observation in the by group otherwise it is 0.