DATA STEP

Data step always starts with key word of data

A data step consist group of statements in SAS language that can read raw data or to create SAS data set.

Data step is useful to perform Data access and Data management.

Purpose of Data step:

→ Retrieving information
→ Checking for errors, validating and correcting SAS code.
→ Create new variables and compute their values
→ Create new data sets from existing SAS datasets
→ Manipulating and reshaping data
→ Generating and printing reports

General group of Statements are Underlined in below Program

```
DATA DATASET;
INFILE DATALINES;
INPUT ID NAME$ AGE SEX$ SALARY;
DATALINES;
001 ABC 23 M 23000
002 DEF 25 F 25000
003 XYZ 22 M 21000
;
RUN;

PROC PRINT DATA=DATASET;
RUN;
```


DATA STATEMENT

Begins a DATA step and provides names for output SAS data sets
Dataset names should write according to SAS Naming rules.

Dataset Options

KEEP:-

Specifies variables for processing or for writing to output SAS data sets

Syntax: - KEEP=variable<s>

Examples: -

```
Data ds1 (keep=name sal);
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds1a (keep=name sal);
Set ds;
Run;
```

```
Data ds1b (keep=name sal);
Infile "C:\Documents and Settings\stansys\Desktop\sample.txt";
Input id name$ sex$ age sal;
Run;
```

```
Proc import datafile ="C:\Documents and Settings\stansys\Desktop\sample.xls"
            Out=work.ds(keep=name sal)
            dbms=excel
            replace;
Run;
```

DROP: -

Excludes variables from output SAS data sets

Syntax: DROP variable<s>

Examples: -

```
Data ds2 (Drop=name sal);
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds2a (drop=name sal);
Set ds;
Run;

Data ds2b (drop=name sal);
Infile "C:\Documents and Settings\stansys\Desktop\sample.txt";
Input id name$ sex$ age sal;
Run;

Proc import data file="C:\Documents and Settings\stansys\Desktop\sample.csv"
            Out=work.ds(drop=name sal)
            dbms=csv
            replace;
Run;
```

RENAME: -

Specifies new names for variables in output SAS data sets

Syntax: -

RENAME= (old-name-1=new-name-1 . . . <old-name-N=new-nameN>);

Examples: -

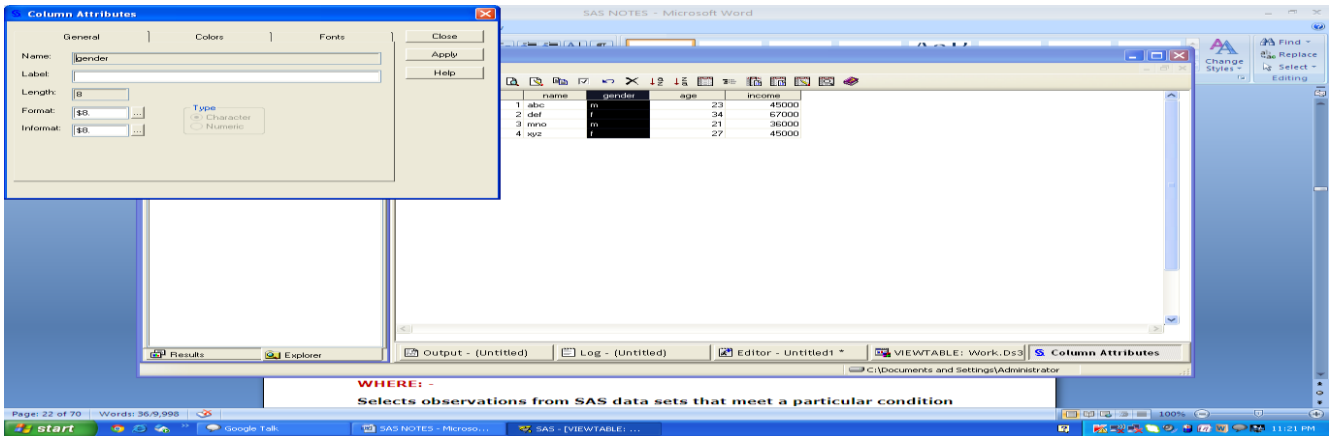
```
Data ds3 (rename=(sex=gender sal=income));
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;

Data ds3 (rename=(sex=gender sal=income));
Infile "C:\Documents and Settings\Stansys\Desktop\sample.txt";
Input id name$ sex$ age sal;
Run;

Data ds;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;

Data ds3a (rename=(sex=gender sal=income));
Set ds;
Run;
```

```
Proc import data file="C:\Documents and Settings\stansys\Desktop\sample.csv"
            Out=work.ds (rename=(sex=gender sal=income))
            dbms=csv
            replace;
Run;
```



Rename changes variable name permanently

WHERE: -

Selects observations from SAS data sets that meet a particular condition

Syntax: - where= (Arguments)

Examples: -

```
Data ds4 (where= (sex='m'));
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds4a (where= (age>=23));
Set ds;
Run;
```

```
Data ds4b (where= (age=23 and sex='m'));
Set ds;
Run;

Data ds4c (where= (id in (001,004)));
Infile "C:\Documents and Settings\stansys\Desktop\sample.txt";
Input id name$ sex$ age sal;
Run;

Proc import data file="C:\Documents and Settings\stansys\Desktop\sample.xls"
            Out=work.ds(where=(sex='f'))
            dbms=excel replace;
Run;
```

REPLACE: -

Controls replacement of like-named temporary or permanent SAS data sets

When we are creating dataset with any name that dataset already is exist in our SAS library, by default it will replace on first dataset when second data step executes, but when don't want to replace use replace=No

Default replace=Yes.

Syntax: - REPLACE=NO | YES

Examples: -


```
Data ds5 (where= (sex='m'));
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;

Data ds5 (replace=no);
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

But in proc import if already dataset is exist we should use replace option to overwrite on existing dataset.

```
Proc import datafile ="C:\Documents and Settings\stansys\Desktop\SAMPLE.csv"
            Out=work.ds dbms=csv;
Run;
```

```
Proc import datafile ="C:\Documents and Settings\stansys\Desktop\SAMPLE.csv"
            Out=work.ds dbms=csv replace;
Run;
```

PW=PASSWORD: -

It's assigns a password to dataset.

Syntax: - PW=Password

Examples: -

```
Data ds6 (pw=sasadm);
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds6b (pw=sasadm);
Infile "C:\Documents and Settings\stansys\Desktop\sample.txt";
Input id name$ sex$ age sal;
Run;
```



```
Proc import data file="C:\Documents and Settings\stansys\Desktop\sample.csv"
            Out=work.ds(pw=krishna)
            dbms=csv
            replace;
Run;
```

Password should max up to 8 characters length it should contains Alphabetic, under square and numbers but not any special characters, and numbers also not first.

Label: -

To assign the label name to data set.

Syntax: Label=Name

Examples: -

```
Data ds7 (Label=sample);
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

NOLIST:-

Suppresses the output of all variables to the SAS log when the value of _ERROR_ is 1.

Syntax: Data dataset**/Nolist**

```
Data ds8/nolist;
Infile datalines;
Input id name sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

It shows like below in log but it won't show _Error_=1 value in log.

```
NOTE: Invalid data for name in line 25 5-7.
NOTE: NOLIST option on the DATA statement suppressed output of variable listing.
NOTE: Invalid data for name in line 26 5-7.
NOTE: NOLIST option on the DATA statement suppressed output of variable listing.
NOTE: Invalid data for name in line 27 5-7.
NOTE: NOLIST option on the DATA statement suppressed output of variable listing.
NOTE: Invalid data for name in line 28 5-7.
NOTE: NOLIST option on the DATA statement suppressed output of variable listing.
NOTE: The data set WORK.DS8 has 4 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```


Debug:-

Enables you to debug your program interactively by helping to identify logic errors, and sometimes data errors

It will give two more windows after EDITOR Window like Proc DEBUGGER LOG and Proc DEBUGGER SOURCE

Proc Debugger log contains information of where the data is not reading and Proc Debugger source contain source code.

Syntax: Data dataset**/Debug**

```
Data ds9/ debug;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

PGM:-

Storing and Executing a Compiled Program

Syntax: Data dataset**/pgm=dataset**

Example: -

```
Data ds10 / pgm=work.ds1;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

The DATA step produces a stored compiled program named Work.DS1

```
Data pgm=WORK.DS1;
Run;
```

VIEW:-

Names a view that the DATA step uses to store the input DATA step view. View-name must match one of the data set names

Syntax: Data dataset**/view=name_of_the_view**

Examples: -

```
Data ds11/ view=ds11;
Set sashelp.class;
Run;

Data ds12/ view=ds12 (source=save);
Set sashelp.class;
Run;

Data ds13/ view=ds13 (source=encrypt);
Set sashelp.class;
Run;
```

```
WARNING: An ALTER / PROTECT password is strongly recommended
         for DATA STEP stored programs and views with encrypted source lines.
```

```
Data ds14/ view=ds14 (alter=sasadm source=encrypt);
Set sashelp.class;
Run;
```

A SAS view is a type of SAS data set that retrieves data values from other files. A SAS view contains only descriptor information such as the data types and lengths of the variables (columns), plus information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. SAS views are of member type VIEW. In most cases, you can use a SAS view as if it were a SAS data file.

- Instead of using multiple DATA steps to merge SAS data sets by common variables, you can construct a SAS view that performs a multi-table join.

- You can save disk space by storing a SAS view definition, which stores only the instructions for where to find the data and how it is formatted, not the actual data.
- SAS views can ensure that the input data sets are always current because data is derived from SAS views at execution time.

SOURCE=source-option
Specifies one of the following source options:
SAVE: Saves the source code that created a stored compiled DATA step program or a DATA step view.

ENCRYPT: Encrypts and saves the source code that created a stored compiled DATA step program or a DATA step view.
Tip: If you encrypt source code, use the ALTER password option as well. SAS issues a warning message if you do not use ALTER.
NOSAVE: Does not save the source code.
Note: If you use the NOSAVE option for a DATA step view, the view cannot be migrated or copied from one version of SAS to another version.

DESCRIBE statement in a DATA step view is useful to write a copy of the source code to the SAS log.

```
Data view=work.ds11;
Describe;
Run;
```

The following example executes a stored compiled DATA step program. It uses the DESCRIBE statement to write a copy of the source code to the SAS log.

```
Data ds15/pgm=dss15;
Set sashelp.class;
Run;

Data pgm=work.dss15;
Describe;
Execute;
Run;
```


Difference between Dataset & View

While the terms "SAS data files" and "SAS views" can often be used interchangeably, here are a few differences to consider:

The main difference is where the values are stored.
A SAS data file is a type of SAS data set that contains both descriptor information about the data and the data values. SAS views contain only descriptor information and instructions for retrieving data that is stored elsewhere. Once the data is retrieved by SAS, it can be manipulated in a DATA step.

A data file is static; a SAS view is dynamic.
When you reference a data file in a later PROC step, you see the data values as they were when the data file was created or last updated. When you reference a SAS view in a PROC step, the view executes and provides you with an image of the data values as they currently exist, not as they existed when the view was defined.
SAS data files can be created on tape or on any other storage medium.

SAS views cannot be stored on tape. Because of their dynamic nature, SAS views must derive their information from data files on random-access storage devices, such as disk drives. SAS views cannot derive their information from files stored on sequentially accessed storage devices, such as tape drives.

SAS views are read only.
You cannot write to a SAS view, but some SQL views can be updated.

SAS data files can have an audit trail.
The audit trail is an optional SAS file that logs modifications to a SAS data file. Each time an observation is added, deleted, or updated, information is written to the audit trail about who made the modification, what was modified, and when.

SAS data files can have generations.
Generations provide the ability to keep multiple copies of a SAS data file. The multiple copies represent versions of the same data file, which is archived each time it is replaced.

SAS data files can have integrity constraints.
When you update a SAS data file, you can ensure that the data conforms to certain standards by using integrity constraints. With SAS views, you can assign integrity constraints to the data files that the views reference.

SAS data files can be indexed.
Indexing might enable SAS to find data in a SAS data file more quickly. SAS views cannot be indexed.

SAS data files can be encrypted.
Encryption provides an extra layer of security to physical files. SAS views cannot be encrypted.

SAS data files can be compressed.
Compression makes it possible to store physical files in less space. SAS views cannot be compressed. The following figure illustrates native and interface SAS data files and their relationship to SAS views.

IN= Data Set Option

Creates a Boolean variable that indicates whether the data set contributed data to the current observation. If BY value is contributed in datset it assigns value as 1.
If BY value is not contributed in datset it assigns value as 0.

Syntax:- IN=VARIABLE

```
Data ds;
Set sashelp.class (in=x);
By sex;
Put x;      *observe in log
Run;

Data patdata;
Infile Datalines;
Input p_id trt_code$;
Datalines;
101 A
102 A
103 B
104 B
;
Run;
Proc sort Data=patdata;
By p_id;
Run;
Data adverse;
Infile Datalines;
Input p_id event$;
Datalines;
101 headaches
107 fever
103 fracture
109 nausea
;
Run;
Proc sort Data=adverse;
By p_id;
Run;
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
/*If a;*/
If a then output;
Put a b;    *observe in log
Run;
```

This options works with SET, MERGE, UPDATE, or MODIFY statements. Not with Datalines.

Creating Multiple Data sets in single data step

We can create multiple SAS datasets from one data step and can use data statement options like below

Syntax: Data dataset1 dataset 2 dataset3;

```
Data ds16 ds17;
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds18 (KEEP=ID NAME SAL) ds19(WHERE=(SEX='f'));
Infile datalines;
Input id name$ sex$ age sal;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```


```
Data ds20 ds21;
Infile datalines;
Input id name$ sex$ age sal;
If sex='f' then output ds20;
else output ds21;
Datalines;
001 abc m 23 45000
002 def f 34 67000
003 mno m 21 36000
004 xyz f 27 45000
;
Run;
```

```
Data ds1 ds2 ds3 ds4 ds5 ds6;
Set sashelp.class;
If age=11 then output ds1;
Else if age=12 then output ds2;
Else if age=13 then output ds3;
Else if age=14 then output ds4;
Else if age=15 then output ds5;
Else if age=16 then output ds6;
Run;
```

Null

The data set name _null_ is reserved for a special purpose here no data set will be create but the programming step executes.

Instead of dataset name we can use _null_.

By using this we can save space in SAS Server

```
DATA _NULL_;
INFILE DATALINES;
INPUT ID NAME$ SEX$ AGE SAL;
DATALINES;
001 ABC M 20 2500
002 DEF M 22 3000
003 XYZ F 21 5000
;
RUN;
```

Use Put statement to see the variable information in log

Put Statement:-

It will write information in SAS log. (or)
It will write information in an external file with file statement.

Syntax: Put Variable(S)

```
Put Variable1 Variable2 Variable3;
Put Variable1= Variable2= Variable3=;
Put @1 Variable1  @10 Variable2  @20 Variable3;
Put @1 Variable1= @10 Variable2=  @20 Variable3=;

Put Variable1;
Put Variable2;
Put Variable3;

Put Variable1=;
Put Variable2=;
Put Variable3=;
```

```
DATA _NULL_;
INFILE DATALINES;
INPUT ID NAME$ SEX$ AGE SAL;
PUT ID NAME SEX AGE SAL;
/*PUT ID= NAME= SEX= AGE= SAL=;*/
/*PUT @1 ID @10 NAME  @20 SEX  @30 AGE @40SAL;*/
DATALINES;
001 ABC M 20 2500
002 DEF M 22 3000
003 XYZ F 21 5000
;
RUN;
```

File Statement:-

It writes the information from SAS to external file.

Syntax: File file-specification <options>

```
DATA DS;
INFILE DATALINES;
INPUT ID NAME$ SEX$ AGE SAL;
DATALINES;
001 ABC M 20 2500
002 DEF M 22 3000
003 XYZ F 21 5000
;
RUN;

DATA _NULL_;
SET DS;
FILE "C:\Documents and Settings\Administrator\Desktop\sas\sample.rtf ";
PUT @1 ID @5 NAME @10 SEX @14 AGE @18 SAL;
RUN;
```

We can create reports in TXT or RTF format using _null_, File & Put statements. See below example for null reporting

```
Data Health;
Infile Datalines;
Input idno 1-4 name $ 6-24 team $ strtwght endwght;
Loss=strtwght-endwght;
Datalines;
1023 David Shaw           red    189 165
1049 Amelia Serrano       yellow 145 124
1219 Alan Nance           red    210 192
1246 Ravi Sinha           yellow 194 177
1078 Ashley McKnight      red    127 118
1221 Jim Brown            yellow 220   .
1095 Susan Stewart        blue   135 127
1157 Rose Collins         green  155 141
1331 Jason Schock         blue   187 172
1067 Kanoko Nagasaka      green  135 122
1251 Richard Rose         blue   181 166
1192 Charlene Armstrong   yellow 152 139
1352 Bette Long           green  156 137
1262 Yao Chen             blue   196 180
1124 Adrienne Fink        green  156 142
1197 Lynne Overby         red    138 125
1133 John VanMeter        blue   180 167
1328 Hisashi Ito          red    155 142
1243 Deanna Hicks         blue   134 122
```

```
1177 Holly Choate        red    141 130
1259 Raoul Sanchez       green  189 172
1017 Jennifer Brooks     blue   138 127
1099 Asha Garg           yellow 148 132
;
Run;

Data _null_;
d=today ();
t=time ();
File 'E:\SAS\TARGET_DATA\DATASTEP_REPORT\Weight_Club4.rtf'
Linesize=200;
Put' ';
Put ' ';
Put @2 'REQUEST # I11796-S'
    @35 'STANSYS SOFTWARE SOLUTIONS'
    @70 "PAGE 1 ";
Put @2 'RUN DATE:' d ddmmyys10.
     @34 'INFORMATION CENTER REQUEST'
     @70 'RUNTIME:' t time8.
Put ' ';
Put ' ';
Run;

Data _null_;
File 'E:\SAS\TARGET_DATA\DATASTEP_REPORT\Weight_Club4.rtf'
Linesize=200 mod;
Put @2 'EMP_ID' @10 'EMP_NAME' @30 'TEAM' @42 'STRTWGHT' @55 'ENDWGHT';
PUT ' ';
Run;

Data _null_;
Set My_SAS.wghtclub;
File 'E:\SAS\TARGET_DATA\DATASTEP_REPORT\Weight_Club4.rtf'
Linesize=200 mod;
Put @2 idno @10 name @30 team
     @42 strtwght @55 endwght;
Run;

Data _null_;
File 'E:\SAS\TARGET_DATA\DATASTEP_REPORT\Weight_Club4.rtf'
Linesize=200 mod;
Put;
Put;
Put;
Put @2 '************ END OF REPORT ************************';
Put @2 '********** GENERATED BY Mr.Krishna ******************';
Run;
```

Interview Questions

Q1) What is SAS Program? Explain Components of SAS Program?

Q2) What is DATA STEP? Explain each purpose of datastep with example?
Q3) What are the minimum statements are required to write Datastep Program?
Q4) What is Data statement?
Q5) What is Syntax of Data statement?
Q6) What are the dataset options do you know?
Q7) What is KEEP Option? Syntax? Example?
Q8) What is DROP Option? Syntax? Example?
Q9) What is RENAME Option? Syntax? Example?
Q10) What is LABEL Option? Syntax? Example?
Q11) Difference between RENAME Option & LABEL Statement?
Q12) Difference between Rename option & Label option?
Q13) What is WHERE Option? Syntax? Example?
Q14) What is PW Option? Syntax? Example?
Q15) What is REPLACE Option? Syntax? Example?
Q16) What is REPLACE Option? Syntax? Example?
Q17) What is IN Option? Syntax? Example?
Q18) What is SORTEDBY Option? Syntax? Example?
Q19) What is _null_?
Q20) Difference between KEEP Statement and KEEP Option?
Q21) Difference between DROP Statement and DROP Option?
Q22) Difference between WHERE Statement and WHERE Option?
Q23) Difference between REPLACE Statement and REPLACE Option?
Q24) Difference between RENAME Statement and RENAME Option?
Q25) Difference between Dataset & View?
Q26) How can you create views ?
Q27) Data is available in the location of "E:\SAS\IQ\DATA STATEMENT"
with the name of DEMOGRAPHIC_DATA in both csv format and notepad
read that data into SAS and write the program for each task ?
tasks are available at TASK notepad in above same location?


STANSYS
SOFTWARE SOLUTIONS