COMBINING DATASETS

Ways to combine datasets

-> Concatenation
-> Interleaving
-> Merge
-> Update
-> Modify

Concatenation:

Combining two or more SAS Datasets into a single SAS Dataset one after other using SET Statement.

The number of observations in new dataset is equal to the sum of the number observations from original datasets.

The second dataset observations followed by the first dataset observations.

If original datasets contain same variables, the variables in new dataset are same as the variables in the original datasets.

If original datasets contain different variables observations from one dataset having missing values for variables in new datasets.

If original datasets contain different data types for variables, concatenation won't happen, need to convert both dataset variables in one data type before performing concatenation.

If original datasets contain different lengths for variables, concatenation done.
The length for new dataset variable is equal to the first set dataset variable length.

Syntax: - Set dataset(s);

Examples:-

If original datasets contain same variables, the variables in new dataset are same as the variables in the original datasets.

```
Data ds1;
Infile datalines;
Input P_id Drug_name$ Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011
102 asp-10mg 14Jan2011
101 asp-05mg 18Jan2011
102 asp-10mg 12Jan2011
101 asp-05mg 21Jan2011
103 asp-15mg 12Jan2011
101 asp-05mg 30Jan2011
102 asp-10mg 12Jan2011
101 asp-05mg 23Jan2011
102 asp-10mg 12Jan2011
;
Run;
```

```
Data ds2;
Infile datalines;
Input P_id Drug_name$ Visit_date;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011
103 asp-15mg 12Jan2011
101 asp-05mg 15Jan2011
104 asp-20mg 12Jan2011
101 asp-05mg 16Jan2011
102 asp-10mg 12Jan2011
101 asp-05mg 17Jan2011
103 asp-15mg 12Jan2011
103 asp-15mg 12Jan2011
101 asp-05mg 15Jan2011
;
Run;
Data ds3;
Set ds1 ds2;
Run;
```

If original datasets contain different variables, observations from one dataset having missing values for variables in new datasets.

```
Data ds1;
Infile Datalines;
Input P_id Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011 m
102 asp-10mg 14Jan2011 f
101 asp-05mg 18Jan2011 f
102 asp-10mg 12Jan2011 f
101 asp-05mg 21Jan2011 m
103 asp-15mg 12Jan2011 m
101 asp-05mg 30Jan2011 f
102 asp-10mg 12Jan2011 m
101 asp-05mg 23Jan2011 f
102 asp-10mg 12Jan2011 f
;
Run;
Data ds2;
Infile Datalines;
Input P_id Drug_name$ Visit_date;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011
```

```
103 asp-15mg 12Jan2011
101 asp-05mg 15Jan2011
104 asp-20mg 12Jan2011
101 asp-05mg 16Jan2011
102 asp-10mg 12Jan2011
101 asp-05mg 17Jan2011
103 asp-15mg 12Jan2011
103 asp-15mg 12Jan2011
101 asp-05mg 15Jan2011
;
Run;
Data ds3;
Set ds1 ds2;
Run;
Data ds1;
Input P_id Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011 m
102 asp-10mg 14Jan2011 f
101 asp-05mg 18Jan2011 f
102 asp-10mg 12Jan2011 f
101 asp-05mg 21Jan2011 m
103 asp-15mg 12Jan2011 m
101 asp-05mg 30Jan2011 f
102 asp-10mg 12Jan2011 m
101 asp-05mg 23Jan2011 f
102 asp-10mg 12Jan2011 f
;
Run;
Data ds2;
Input P_id Drug_name$ Visit_date Age;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011 34
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 23
104 asp-20mg 12Jan2011 28
101 asp-05mg 16Jan2011 21
102 asp-10mg 12Jan2011 30
101 asp-05mg 17Jan2011 28
103 asp-15mg 12Jan2011 23
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 25
;
Run;
```

```
Data ds3;
Set ds1 ds2;
Run;
```

If original Datasets contain different Data types for variables, concatenation won't happen

```
Data ds1;
Infile Datalines;
Input P_id$ Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011 m
102 asp-10mg 14Jan2011 f
101 asp-05mg 18Jan2011 f
102 asp-10mg 12Jan2011 f
101 asp-05mg 21Jan2011 m
103 asp-15mg 12Jan2011 m
101 asp-05mg 30Jan2011 f
102 asp-10mg 12Jan2011 m
101 asp-05mg 23Jan2011 f
102 asp-10mg 12Jan2011 f
;
Run;
Data ds2;
Infile Datalines;
Input P_id Drug_name$ Visit_date Age;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011 34
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 23
104 asp-20mg 12Jan2011 28
101 asp-05mg 16Jan2011 21
102 asp-10mg 12Jan2011 30
101 asp-05mg 17Jan2011 28
103 asp-15mg 12Jan2011 23
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 25
;
Run;
Data ds3;
Set ds1 ds2;
Run;
```

Above program having ERROR: Variable p_id has been defined as both character and numeric.

So use Input function to convert **P_id** from character to numeric Data type before performing concatenation.

```
Data ds1;
Infile Datalines;
Input P_id$ Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011 m
102 asp-10mg 14Jan2011 f
101 asp-05mg 18Jan2011 f
102 asp-10mg 12Jan2011 f
101 asp-05mg 21Jan2011 m
103 asp-15mg 12Jan2011 m
101 asp-05mg 30Jan2011 f
102 asp-10mg 12Jan2011 m
101 asp-05mg 23Jan2011 f
102 asp-10mg 12Jan2011 f
;
Run;
```

Converting P_id from character no numeric

```
Data ds1a (drop=p_id rename=(p_id1=p_id));
Set ds1;
p_id1=Input (p_id, 3.);
Run;

Data ds2;
Infile Datalines;
Input P_id Drug_name$ Visit_date Age;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011 34
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 23
104 asp-20mg 12Jan2011 28
101 asp-05mg 16Jan2011 21
102 asp-10mg 12Jan2011 30
101 asp-05mg 17Jan2011 28
103 asp-15mg 12Jan2011 23
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 25
;
Run;

Data ds3;
Set ds1a ds2;
Run;
```

If original Datasets contain different lengths for variables, concatenation done. The length for new Dataset variable is equal to the first Dataset variable length in SET Statement..

```
Data ds1;
Infile Datalines;
Input P_id P_Name:$10. Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 Alfred asp-05mg 11Jan2011 m
101 Alfred asp-05mg 12Jan2011 m
102 Henrycarol asp-10mg 12Jan2011 f
102 Henrycarol asp-10mg 14Jan2011 f
101 Alfred asp-05mg 15Jan2011 m
101 Alfred asp-05mg 18Jan2011 m
102 Henrycarol asp-10mg 18Jan2011 m
101 Alfred asp-05mg 21Jan2011 m
101 Alfred asp-05mg 23Jan2011 m
101 Alfred asp-05mg 30Jan2011 m
102 Henrycarol asp-10mg 20Jan2011 f
;
Run;

Data ds2;
Infile Datalines;
Input P_id P_Name$ Drug_name$ Visit_date date9. sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
103 John asp-15mg 12Jan2011 m
103 John asp-15mg 15Jan2011 m
104 Louise asp-20mg 12Jan2011 f
103 John asp-15mg 18Jan2011 m
103 John asp-15mg 22Jan2011 m
;
Run;

Data ds3a;
Set ds1 ds2;
Run;
```

But second dataset variable length is more than first dataset variable length there is a chance to lose few characters from second dataset so use length statement before set statement.

```
Data ds3b;
/*Length P_Name$10.;*/
Set ds2 ds1;
Run;
```

Concatenation with options (SET Statement Options)

```
Data ds1;
Infile Datalines;
Input P_id Drug_name$ Visit_date Sex$;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011 m
102 asp-10mg 14Jan2011 f
101 asp-05mg 18Jan2011 f
102 asp-10mg 12Jan2011 f
101 asp-05mg 21Jan2011 m
103 asp-15mg 12Jan2011 m
101 asp-05mg 30Jan2011 f
102 asp-10mg 12Jan2011 m
101 asp-05mg 23Jan2011 f
102 asp-10mg 12Jan2011 f
;
Run;
Data ds2;
Infile Datalines;
Input P_id Drug_name$ Visit_date Age;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 11Jan2011 34
103 asp-15mg 12Jan2011 32-
101 asp-05mg 15Jan2011 23
104 asp-20mg 12Jan2011 28
101 asp-05mg 16Jan2011 21
102 asp-10mg 12Jan2011 30
101 asp-05mg 17Jan2011 28
103 asp-15mg 12Jan2011 23
103 asp-15mg 12Jan2011 32
101 asp-05mg 15Jan2011 25
;
Run;
```

Firstobs and Obs

```
Data ds3;
Set ds1(firstobs=4) ds2;
Run;

Data ds3;
Set ds1(firstobs=4) ds2(obs=7);
Run;

Data ds3;
Set ds1 ds2(firstobs=4 obs=8);
Run;
```

Keep, Drop, Rename

```
Data ds3;
Set ds1(keep=drug_name visit_date) ds2;
Run;
Data ds3;
Set ds1(keep=drug_name visit_date) ds2(rename=(p_id=patient_id));
Run;
Data ds3;
Set ds1(drop=drug_name visit_date) ds2(rename=(p_id=patient_id));
Run;
```

Point=Slice

We can use this option for selecting particular observations from dataset with less time.

```
Data ds1;
Infile Datalines;
Input P_id Drug_name$ Visit_date ;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 12Jan2011
102 asp-10mg 14Jan2011
103 asp-10mg 12Jan2011
104 asp-05mg 21Jan2011
105 asp-15mg 12Jan2011
106 asp-10mg 12Jan2011
107 asp-15mg 12Jan2011
;
Run;
```

Selecting only one specific observation

```
Data ds2;
Slice=2;
Set ds1 point=slice;
Output;
Stop;
Run;
```

Selecting specific multiple observations

```
Data ds3;
Do slice=2,4,5;
Set ds1 point=slice;
Output;
End;
Stop;
Run;
```

Another way in SAS Datastep

```
Data ds;
Set SASHELP.CLASS;
If _n_=3;
/*If _n_ in(2,4,5);*/
Run;
```

Another way in Proc sql

Selecting only one specific observation

```
proc sql;
create table stansys as select * from sashelp.class where monotonic()=10;
Quit;
```

Selecting multiple specific observation

```
Proc sql;
Create table stansys as select * from sashelp.class where monotonic()  in
(1,3,4,6,8,10,15);
Quit;
```

NOBS=NOBS

We can use this option for identifying total no of obs is there in dataset.

Entire dataset in reverse order.

```
Data ds;
Do i=nobs to 1 by -1;
Set SASHELP.CLASS point=i nobs=nobs;
output;
end;
stop;
Run;
```

Selecting 1st record for every 3 records.

```
Data ds;
Do i=1 to nobs by 3;
Set SASHELP.CLASS point=i nobs=nobs;
output;
end;
stop;
Run;
```

Selecting 1st record for every 3 records in reverse order of dataset.

```
Data ds;
Do i=nobs to 1 by -3;
Set SASHELP.CLASS point=i nobs=nobs;
output;
end;
stop;
Run;
```

Selecting every alternative record from first.

```
Data ds;
Do i=1 to nobs by 2;
Set SASHELP.CLASS point=i nobs=nobs;
output;
end;
stop;
Run;
```

Selecting every 3rd record.

```
%let n=3;
Data ds;
Set sashelp.class;
if mod(_n_,&n) eq 0 then output;
Run;
```

Concatenation with multiple SET statements (one to one reading)

Combines observations from two or more SAS Datasets into a one observation using two or more SET statements. The new Dataset contains all the variables from all Input Datasets.

Syntax:-

```
Set dataset1;
Set dataset2;
Set datasetN;
```

Both datasets contains same variables and same number of observations

The number of observations in the new Dataset is the number of observations from last set statement Dataset if the variables are same from both datasets.

```
Data ds1;
Infile Datalines;
Input a b c;
Datalines;
1 2 3
4 5 6
;
Run;
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
Data ds3;
Set ds1;
Set ds2;
Run;
```

In above example last dataset variables (a b c) are same with first dataset variables so last dataset variables overwrite on first dataset variables.

So output dataset contains data from last set statement dataset.

Both datasets contains different variables and same number of observations

If the Dataset doesn't contains common variables extra variables which are not common also comes into output Dataset. And common variables are overwrite by second dataset

```
Data ds1;
Infile Datalines;
Input a b c d;
Datalines;
1 2 3 4
4 5 6 7
;
Run;

Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
Data ds3;
Set ds1;
Set ds2;
Run;
```

In above example last dataset variables (a b c) are same with first dataset Variables so it's overwrite from last set statement data set and d variable comes from first dataset. Here same variables over write from last dataset and extra variables comes from any one dataset. **Using this we can perform Merge.**

Observe below example how it performs Merge.

```
Data ds1;
Infile datalines;
Input empid empname$ city$;
Datalines;
1 a hyd
2 b hyd
3 c hyd
;
Run;

Data ds2;
Infile datalines;
Input empid sales;
Datalines;
1 20
```

```
2 30
3 40
;
Run;
Data ds3;
Set ds1;
Set ds2;
Run;
```

Both datasets contains same variables and different number of observations

Both datasets contain same variables so last set statement dataset values are overwrite on first dataset but no of observations are different so which ever dataset contains less observations that many observations come from last set statement dataset.

```
Data ds1;
Infile Datalines;
Input a b c ;
Datalines;
1 2 3
4 5 6
7 8 9
;
Run;
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
Data ds3;
Set ds1;
Set ds2;
Run;
```

In above example last dataset ds2 variables (a b c) are same with first dataset ds1variables so a b c variables overwrite from ds2 dataset it means observations comes from last dataset but that no is equal to least count of observations from any one dataset.

```
Data ds3;
Set ds2;
Set ds1;
Run;
```

In this example last dataset (ds1) variables a b c are same with first dataset ds2 so a b c variable come from last dataset ds1. And observations come from last dataset that is equal to the least no of observations from any one dataset.

Both datasets contains different variables and different number of observations

If first dataset contains more observations and both datasets contains different variables second dataset overwrite on first dataset values and unmatched variables also comes in output dataset

But second dataset contains more observations and both datasets different variables it will read data from second dataset only lowest number of observations come to output dataset from second dataset and unmatched variables also comes into output dataset

```
Data ds1;
Infile Datalines;
Input a b c ;
Datalines;
1 2 3
4 5 6
7 8 9
;
Run;
```

```
Data ds2;
Infile Datalines;
Input a b c d;
Datalines;
3 4 5 6
6 7 8 9
;
Run;
```

```
Data ds3;
Set ds1;
Set ds2;
Run;
```

In above example the entire data comes from last dataset why because last dataset variables a b c is same with first dataset so it over writes and least no of observations also from last dataset.

```
Data ds3;
Set ds2;
Set ds1;
Run;
```

In above example 2 records come from second dataset ds1 and d variable comes from first dataset ds2. And 2obs should be there in output

Interleaving:

Use SET statement and BY statement to combine multiple Datasets into single Dataset.

The number of observations in new Dataset is equal to the sum of the number of observations from original Datasets.

The observations in new Dataset are arranged the values of the BY variables.

We can interleave Datasets using BY variable or using Index.

Note: To perform interleave both Datasets variables should be same, same Data types, same length and should be sorting order.

Syntax:-

Set Dataset(s);
By variable(S);

Examples:-

```
Data ds1;
Infile Datalines;
Input P_id Drug_name$ Visit_date ;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 02Jan2011
102 asp-10mg 02Jan2011
101 asp-05mg 10Jan2011
102 asp-10mg 10Jan2011
101 asp-05mg 21Jan2011
103 asp-15mg 02Jan2011
;
Run;
```

```
Data ds2;
Infile Datalines;
Input P_id Drug_name$ Visit_date ;
Informat Visit_date date9.;
Format Visit_date date9.;
Datalines;
101 asp-05mg 30Jan2011
103 asp-15mg 10Jan2011
102 asp-10mg 20Jan2011
104 asp-05mg 02Jan2011
;
Run;
```

```
Data ds3;
Set ds1 ds2;
By p_id;
Run;
ERROR: BY variables are not properly sorted on Data set WORK.DS1.
```

As per syntax rules both Dataset should be in sorting order

```
Proc sort Data=ds1;
By p_id;
Run;

Proc sort Data=ds2;
By p_id;
Run;

Data ds3;
Set ds1 ds2;
By p_id;
Run;
```

When we use a **BY** statement along with **SET** statement (performs interleaving) in Data step SAS create two automatic variables along with raw Data variables those are **First.Variable** and **Last.Variable**

First.Byvariable:-

Value is 1 for the first observation in the by group and value 0 for all other observations in the by group

Last.Byvariable:-

Value is 1 for the last observation in the by group and value 0 for all other observations in the by group

In First.Variable and Last.Variable, Variable means it's a BY Variable name.

P_id	Drug_name	Visit_date	First.p_id	Last.p_id
101	asp-05mg	2-Jan-11	1	0
101	asp-05mg	10-Jan-11	0	0
101	asp-05mg	20-Jan-11	0	0
101	asp-05mg	30-Jan-11	0	1
102	asp-10mg	2-Jan-11	1	0
102	asp-10mg	10-Jan-11	0	0
102	asp-10mg	20-Jan-11	0	1
103	asp-15mg	2-Jan-11	1	0
103	asp-15mg	10-Jan-11	0	1
104	asp-05mg	2-Jan-11	1	1

Using First.Variable and Last.Variable we can perform

To report each group first visiting Information

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=1;
/*If first.p_id=1 then output;*/
Run;
```

To report each group last visiting Information

```
Data ds3;
Set ds1 ds2;
By p_id;
If last.p_id=1;
/*If last.p_id=1 then output;*/
Run;
```

To report each group all visiting Information except first and last

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=0 and last.p_id=0;
Run;
```

To report who ever visited only once

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=1 and last.p_id=1;
Run;
```

To report every first visiting information except who visited once

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=1 and last.p_id=0;
Run;
```

To report every last visiting information except who visited once

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=0 and last.p_id=1;
Run;
```

To report every patient all visiting information except last from multiple visits and who visits once also.

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=1 or last.p_id=0;
Run;
```

To report every patient all visiting information except first from multiple visits and who visits once also.

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=0 or last.p_id=1;
Run;
```

To report every group first and last visiting information.

```
Data ds3;
Set ds1 ds2;
By p_id;
If first.p_id=1 or last.p_id=1;
Run;
```

Banking Example

```
Data ds1;
Infile Datalines;
Input Loan_no$ 1-5  Customer  $7-15 Loan_amt  Loan_date ;
Informat Loan_date date9. Loan_amt dollar5. ;
Format Loan_date date9.  Loan_amt dollar5. ;
Datalines;
LP101 RaviSinha 3000 02Jan2011
LP102 AlanNance 2500 02Jan2011
LP101 RaviSinha 5000 10Jan2011
LP102 AlanNance 1500 10Jan2011
LP101 RaviSinha 4500 20Jan2011
LP103 JimBrown  4500 02Jan2011
;
Run;
Proc sort data=ds1;
By Loan_no;
Run;
Data ds2;
Infile Datalines;
Input Loan_no$ 1-5  Customer  $7-15 Loan_amt  Loan_date ;
Informat Loan_date date9. Loan_amt dollar5. ;
Format Loan_date date9.  Loan_amt dollar5. ;
Datalines;
LP101 RaviSinha $3000 30Jan2011
LP103 JimBrown $2500 10Jan2011
LP102 AlanNance $5000 20Jan2011
LP104 AshleyMcK $1500 01Jan2011
;
Run;
Proc sort data=ds2;
By Loan_no;
Run;
```

```
Data ds3;
Set ds1 ds2;
By Loan_no;
Run;
```

To report each subject first visiting Information

```
Data ds3a;
Set ds1 ds2;
By Loan_no;
If First.Loan_no=1;
Run;
```

To report each subject last visiting Information

```
Data ds3b;
Set ds1 ds2;
By Loan_no;
If Last.Loan_no=1;
Run;
```

To report each subject visiting Information except first and last

```
Data ds3c;
Set ds1 ds2;
By Loan_no;
If First.Loan_no=0 and Last.Loan_no=0;
Run;
```

To report who ever visited only once

```
Data ds3d;
Set ds1 ds2;
By Loan_no;
If First.Loan_no=1 and Last.Loan_no=1;
Run;
```

Execute below programs and see the results

```
Data ds3e;
Set ds1 ds2;
By p_id;
If First.Loan_no=1 and Last.Loan_no=0;
Run;

Data ds3;
Set ds1 ds2;
By p_id;
If First.Loan_no=0 and Last.Loan_no=1;
Run;

Data ds3;
Set ds1 ds2;
By p_id;
If First.Loan_no=1 or Last.Loan_no=0;
Run;
```

```
Data ds3;
Set ds1 ds2;
By p_id;
If First.Loan_no=0 or Last.Loan_no=1;
Run;
```

```
Data ds3;
Set ds1 ds2;
By p_id;
If First.Loan_no=1 or Last.Loan_no=1;
Run;
```

Select every 3rd highest sales on each city wise

```
Data dsn1;
Infile datalines;
Input id name$ sales city$;
Datalines;
1 x 50 hyd
2 a 20 bang
3 c 25 bang
4 y 45 hyd
5 z 25 hyd
6 b 60 bang
7 d 20 hyd
8 e 10 bang
;
Run;
proc sort data=dsn1 ;
By city descending sales;
Run;
Data dsn2;
Set dsn1;
By city;
If first.city then n=0;
n+1;
If n=3 then output;
/*If n=1 then output;*/
/*If n=2 then output;*/
Run;
```

Merge

Joins/Combines observations from two or more SAS Datasets into single observation in new Dataset.

-> Merge usually joins Datasets with different variables.
-> Output Dataset contains all the variables from all Datasets.
-> Upto 100 Datasets can merge in one step.
-> Observations are joining one to one with out BY statement
-> Observations are Match Merge with BY statement.

We can Merge the data in different ways

1) Merge in DATASTEP
2) Merge with SQL
3) Merge with SET-KEY
4) Merge with FORMAT
5) Merge with HASH TABLE
6) Merge with ARRAY
7) Merge with UPDATE
8) Merge with CALL EXECUTE

1) Merge in DATASTEP

➔ Merge with out By statement
➔ Merge with By statement

Merge with out By statement

Combines observations from two or more SAS Datasets into a single observation in a new dataset using the MERGE statement.

Combines first observations from all Datasets into the first observation in new Dataset. The second observations from all Datasets into the second observation in new Dataset etc...

The number of observations in the new Dataset is equal to the maximum number of observations from original Datasets.

Syntax:-

Merge Dataset(s);

Examples:-

Same no of Observations from both Datasets

```
Data ds1;
Input id name$ sex$ address$;
Datalines;
001 abc m bang
002 def m hyd
003 jkl f chen
004 mno f bang
005 xyz m mum
;
Run;
```

```
Data ds2;
Input dob date9. doj:date9. sal ;
Format dob date9. doj date9.;
Datalines;
01Feb1983 12Jan2011 45000
23Mar1983 20Jan2011 50000
12Oct1983 13Feb2011 34000
02Jan1984 19May2011 28000
28Apr1985 11Jun2011 29000
;
Run;
```

```
Data ds3;
Merge ds1 ds2;
Run;
```

Different no of observations from both Datasets

```
Data ds1;
Infile Datalines;
Input id name$ sex$ address$;
Datalines;
001 abc m bang
002 def m hyd
003 jkl f che
004 mno f bang
005 xyz m mum
006 asd f hyd
;
Run;
```

```
Data ds2;
Infile Datalines;
Input dob date9. doj:date9. sal;
Format dob date9. doj date9.;
Datalines;
01Feb1983 12Jan2011 45000
23Mar1983 20Jan2011 50000
12Oct1983 13Feb2011 34000
02Jan1984 19May2011 28000
28Apr1985 11Jun2011 29000
;
Run;
```

```
Data ds3;
Merge ds1 ds2;
Run;
```

Match Merge (Merge with By statement)

Combines observations from two or more SAS Datasets into a single observation in a Dataset according to values of the common variables from both Datasets.

The number of observations in the new dataset is equal to the largest number of observations in each BY group from original Datasets.

Before performing match Merge all Datasets must be sorted based on common variables.

To perform match Merge we can use the MERGE statement with BY statement.

In the SAS match-merge, the matching process is controlled by the BY variables. BY variables are the variables listed in the BY statement.

BY variables should be key variables. Key variables are either character or numeric variables that uniquely identify or label the records or observations within the input datasets.

Syntax:-

Merge Dataset(s);
By variable(s);

Examples:-

```
Data ds1;
Infile Datalines;
Input id name$ sex$ address$;
Datalines;
001 abc m bang
002 def m hyd
003 jkl f che
004 mno f bang
005 xyz m mum
006 asd f hyd
;
Run;

Proc sort Data=ds1;
By id;
Run;

Data ds2;
Infile Datalines;
Input id dob date9. doj:date9. sal ;
Format dob date9. doj date9.;
Datalines;
001 01Feb1983 12Jan2011 45000
004 23Mar1983 20Jan2011 50000
005 12Oct1983 13Feb2011 34000
002 02Jan1984 19May2011 28000
003 28Apr1985 11Jun2011 29000
;
Run;

Proc sort Data=ds2;
By id;
Run;
```

```
Data ds3;
Merge ds1 ds2;
By id;
Run;
```

In above example both ds1 and ds2 datasets contains common variable **id,** based on common variable need to sort both datasets and can perform match merge

```
Data demo;
Infile datalines;
Input name$ 1-25 age 27-28 sex$30;
Datalines;
Vincent, Martina          34 F
Phillipon, Marie-Odile    28 F
Gunter, Thomas            27 M
Harbinger, Nicholas       36 M
Benito, Gisela            32 F
Rudelich, Herbert         39 M
Sirignano, Emily          12 F
Morrison, Michael         32 M
;
Run;
Proc sort data=demo;
By name;
Run;
Data finance;
Infile datalines;
Input ssn$ 1-11 name$ 13-40 salary;
Datalines;
074-53-9892 Vincent, Martina            35000
776-84-5391 Phillipon, Marie-Odile      29750
929-75-0218 Gunter, Thomas              27500
446-93-2122 Harbinger, Nicholas         33900
228-88-9649 Benito, Gisela              28000
029-46-9261 Rudelich, Herbert           35000
442-21-8075 Sirignano, Emily            5000
;
Run;
Proc sort data=finance;
By name;
Run;
Data new;
Merge demo (drop=age) finance;
By name;
Run;
```

Techniques, Tricks, and Traps in Match Merge

A common mistake in Match merge is forget to include the BY statement.

```
Data ds1;
Infile datalines;
Input id$ name$;
Datalines;
A01 SUE
A02 TOM
A05 KAY
A10 JIM
;
Run;
Data ds2;
Infile datalines;
Input id$ age sex$;
Datalines;
A01 58 F
A02 20 M
A04 47 F
A10 11 M
;
Run;
Data ds3;
Merge ds1 ds2;
Run;
```

Even with this simple example, there is already a hint of problems. Observe that the records **A05** (3rd record from ds1) is not there in ds2. So **A04** (3rd record from ds2) is merging with **A05** record. Notice that the A05 ID is lost in this merge and the name Kay is moved from ID=A05 to ID=A04, and one does not even get a note or error I log to say that something is wrong. This kind of merge we getting wrong output so before performing match merge both datasets should be in sorting order and need to use that BY statement with Merge statement like below

```
Proc sort data=ds1;
By id;
Run;
Proc sort data=ds2;
By id;
Run;
Data ds3;
Merge ds1 ds2;
By id;
Run;
```

Both by variables have different lengths

```
Data ds1;
Infile datalines;
Input id $1-3 x 6-7;
Datalines;
A22  12
A38  88
A51  33
;
Run;

Data ds2;
Infile datalines;
Input ID $1-4 y 6-7;
Datalines;
A22  72
A38  31
A41  11
A511 58
;
Run;

Data ds3;
Merge ds1 ds2;
By id;
Run;
```

In above example, the variable, ID, has a LENGTH=3 in the first dataset and a LENGTH=4 in the second dataset. At compile time, the program data vector, for the output file, the attributes of each variable is determined by the first input data set where they appear. Thus, in this case, the after first dataset in the merge statement is scanned, the data vector is (ID $3, x). Then, the second dataset is scanned the data vector is (ID $4, y). And new variables added to the vector, so that the final output data is (ID$3 X Y). Since the ID has a LENGTH=3 in the data vector, the value of ID=511 in the second file is clipped to A51 and matched with the record A51 from the first file. This is an example of how, when the LENGTHs are different, one can get undesired results.

```
Data ds3a;
Merge ds2 ds1;
By id;
Run;
```

In above example, shows how reversing the order of the data sets in the merge statement can sometimes change the values and records in the output file. In this case merging is happening correctly because second dataset is scanning first in pdv so length For id are 4 for both datasets in output so we can get proper results.

Types of Match Merge

1) zero-to-one
2) one-to-zero
3) one-to-one
4) one-to-many
5) many-to-one
6) few-to-many
7) many-to-few
8) many-to-many

One to One / Zero to One / One to Zero Merge

One to One Merge:

One observation from first dataset merges with One observation in second dataset for same by group

Zero to One Merge:

Zero observation from first dataset merges with One observation in second dataset for same by group

One to Zero Merge:

One observation from first dataset merges with Zero observation in second dataset for same by group

```
Data ds1;
Infile datalines;
Input id$ name$;
Datalines;
A01 SUE
A02 TOM
A05 KAY
A10 JIM
;
Run;

Proc sort data=ds1;
By id;
Run;

Data ds2;
Infile datalines;
Input id$ age sex$;
Datalines;
A01 58 F
A02 20 M
A04 47 F
A10 11 M
;
Run;

Proc sort data=ds2;
By id;
Run;
```

```
Data ds3;
Merge ds1 ds2;
By id;
Run;
```

In above example performs one-to-one, zero-to-one, and one-to-zero Match-merge
In ds1 dataset A01 id is merging with A01 from ds2
A02 id is merging with A02 from ds2
A10 id is merging with A10 from ds2 dataset that is **One-to-One Merge**
A05 from ds1 is not merging with ds2 dataset that is **One-to-Zero Merge**
A04 is not there from ds1 but its there from ds2 dataset so it's **Zero-to-One Merge**

Many-to-Many Match Merge:

Within a group Many observation from first dataset merges with Many observation in second dataset.

Internally it is again one to one merge (we can perform Many to Many merge when we have same no of records from both datasets for same by group)

```
Data ds1;
Infile datalines;
Input id$ x;
Datalines;
A25 24
A25 22
A25 76
;
Run;
Data ds2;
Infile datalines;
Input id$ y;
Datalines;
A25 24
A25 22
A25 76
;
Run;
Data ds3;
Merge ds1 ds2;
By id;
Run;
```

The many-to-many match-merge is essentially a one-to-one Merge (Merge with out by) and has the same drawbacks and dangers. Specifically, one has very little control over the actual order of the records within the BY group for each of the input data sets.

For example, how does one know that the first value of x=24 is supposed to be matched with the first value of y=4 Why shouldn't x=24 be matched with y=91 (the second value of y)? If great care is not taken, a many-to-many merge can result in random matching of variable values.

This Many-to-Many merge is dangerous and unreliable sometimes so program has to take care and he has to choose some additional by variables to merge properly

Few-to-Many or Many-to-few Match Merge

Few to Many Merge:

Within a group Few observations from first dataset merges with Many observation in second dataset.

Many to Few Merge:

Within a group Many observations from first dataset merge with Few observations in second dataset.

The few-to-many type of match-merge occurs when for a given BY group, there is more than one record in the first input data set, and the second input data set has more records than the first.

Few to Many merge are combination of one to one and one to many Merge.
Many to Few merge are combination of one to one and many to one Merge.

```
Data ds1;
Infile datalines;
Input id$ x;
Datalines;
A92 70
A92 46
;
Run;

Data ds2;
Infile datalines;
Input id$ Y;
Datalines;
A92 14
A92 72
A92 7
;
Run;

Data ds3a;
Merge ds1 ds2;
By id;
Run;

Data ds3b;
Merge ds2 ds1;
By id;
Run;
```

Few-to-Many or Many-to-Few merge also dangerous like Many-to-Many Merge
To overcome those problems programmer has to choose correct by variables.

One-to-Many/Many-to-One Match Merge

One to Many Merge:

Within a group one observation from first dataset merges with many observations in second dataset.

Many to one Merge:

Within a group many observations from first dataset merges with one observation from second dataset.

The simplest and most useful merge after the one-to-one match-merge is the One-to-many match-merge.

```
Data ds1;
Infile datalines;
Input id$ x;
Datalines;
A32 5
A35 3
;
Run;
Data ds2;
Infile datalines;
Input id$ Y;
Datalines;
A32 15
A32 22
A32 61
;
Run;
Data ds3a;
Merge ds1 ds2;
By id;
Run;
Data ds3b;
Merge ds2 ds1;
By id;
Run;
```

In above example there are two BY groups. The first output record is the same as in a one-to-one match-merge. But for the second record in the ds2 dataset there is no corresponding ds1 record, so SAS retains the x value from the first ds1 record and passes it to the second output record.

Observe how values overwrites in Merge

Both datasets contains same variables and same number of observations

```
Data ds1;
Infile Datalines;
Input a b c;
Datalines;
1 2 3
4 5 6
;
Run;
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
```

```
Data ds3;
Merge ds1 ds2;
Run;
```

```
Data ds3a;
Merge ds1 ds2;
By a;
Run;
```

Both datasets contains different variables and same number of observations

```
Data ds1;
Infile Datalines;
Input a b c d;
Datalines;
1 2 3 4
4 5 6 7
;
Run;
```

```
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
```

```
Data ds3;
Merge ds1 ds2;
Run;
```

```
Data ds3a;
Merge ds1 ds2;
By a;
Run;
```

Both datasets contains same variables and different number of observations

```
Data ds1;
Infile Datalines;
Input a b c ;
Datalines;
1 2 3
4 5 6
7 8 9
;
Run;
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
6 7 8
;
Run;
Data ds3;
Merge ds1 ds2;
Run;
Data ds3a;
Merge ds1 ds2;
By a;
Run;
```

Both datasets contains different variables and different number of observations

```
Data ds1;
Infile Datalines;
Input a b c ;
Datalines;
1 2 3
4 5 6
7 8 9
;
Run;
Data ds2;
Infile Datalines;
Input a b c d;
Datalines;
3 4 5 6
6 7 8 9
;
Run;
```

```
Data ds3;
Merge ds1 ds2;
Run;
Data ds3a;
Merge ds1 ds2;
By a;
Run;
```

Both datasets contains same variables and same number of observations But both datasets contain same by group observation.

```
Data ds1;
Infile Datalines;
Input a b c;
Datalines;
1 2 3
4 5 6
;
Run;
Data ds2;
Infile Datalines;
Input a b c;
Datalines;
3 4 5
4 7 8
;
Run;
Data ds3;
Merge ds1 ds2;
Run;
Data ds3a;
Merge ds1 ds2;
By a;
Run;
```

Left Join in SAS

When we perform left outer join all the Data comes from left table and matching Data comes from right table into output Dataset.

```
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;

/*Thru Data step*/
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
/*If a;*/
If a then output;
Run;

/*Thru SQL*/
Proc sql;
Create table pat_adverse as
Select a.*, b.* from patdata a left outer join adverse b on a.p_id=b.p_id;
Quit;
```

Right Join in SAS

When we perform right outer join all the Data comes from right table and matching Data comes from left table into output Dataset.

```
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;

/*Thru Data step*/
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
/*If b;*/
If b then output;
Run;

/*Thru SQL*/
Proc sql;
Create table pat_adverse as
Select b.*, a.* from patdata a right outer join adverse b on b.p_id=a.p_id;
Quit;
```

Inner Join in SAS

When we perform inner join all matching Data comes from both Datasets into output Dataset.

```
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;

/*Thru Data step*/
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
/*If a and b;*/
If a and b then output;
Run;
/*Thru SQL*/
Proc sql;
Create table pat_adverse as
Select a.*, b.* from patdata a, adverse b where a.p_id=b.p_id;
Quit;
Proc sql;
Create table pat_adverse as
Select a.*, b.* from patdata a inner join adverse b on a.p_id=b.p_id;
Quit;
```

Full Join in SAS

When we perform full outer join all the Data comes from all the tables into output Dataset.

```
Data patdata;
Infile Datalines;
Infile 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;
Infile p_id event$;
Datalines;
101 headaches
107 fever
103 fracture
109 nausea
;
Run;

Proc sort Data=adverse;
By p_id;
Run;

/*Thru Data step*/
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
/*If a or b;*/
If a or b then output;
Run;

/*Thru SQL*/
Proc sql;
Create table pat_adverse as
Select a.*, b.* from patdata a full outer join adverse b on a.p_id=b.p_id;
Quit;
```

Observe below programs results how it is merging the values.

```
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 not a and b;
Run;
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
if a and not b;
Run;
Data pat_adverse;
Merge patdata(in=a) adverse(in=b);
By p_id;
if a ne b;
Run;
```

UPDATE

To apply the changes in one Dataset with Information of another Dataset.

To perform update we need two Datasets

1) Master Dataset: Contains original Information.

2) Transaction Dataset: Contains changing Information.

Before performing update both Datasets should be in sorting order

Examples:-

```
Data ds1;
Infile Datalines;
Input id name$ sex$ sal;
Datalines;
001 abc m 5000
003 jkl f 6000
002 def m 6000
005 xyz m 5000
004 mno f 6000
006 asd f 8000
;
Run;
Proc sort Data=ds1;
By id;
Run;
Data ds2;
Infile Datalines;
Input id sal;
Datalines;
001 7500
006 9000
002 8000
005 8000
004 7900
003 6500
;
Run;
Proc sort Data=ds2;
By id;
Run;
Data ds3;
Update ds1 ds2;
By id;
Run;
Data master;
Infile datalines;
Input id 1-8 name $ 9-27 street $ 28-47 city $ 48-62 state $ 63-64 zip $ 67-71;
```

```
Datalines;
1001    Ericson, Jane      111 Clancey Court   Chapel Hill    NC  27514
1002    Dix, Martin        4 Shepherd St.      Norwich        VT  05055
1003    Gabrielli, Theresa 24 Ridgetop Rd.     Westboro       MA  01581
1004    Clayton, Aria      14 Bridge St.       San Francisco  CA  94124
1005    Archuleta, Ruby    Box 108             Milagro        NM  87429
1006    Misiewicz, Jeremy  43-C Lakeview Apts. Madison        WI  53704
1007    Ahmadi, Hafez      5203 Marston Way    Boulder        CO  80302
1008    Jacobson, Becky    1 Lincoln St.       Tallahassee    FL  32312
1009    An, Ing            95 Willow Dr.       Charlotte      NC  28211
1010    Slater, Emily      1009 Cherry St.     York           PA  17407
;
Run;
Data Trans;
Infile datalines;
Input id 1-8 name $ 9-27 street $ 28-47 city $ 48-62 state $ 63-64 zip $ 67-71;
Datalines;
1002    Dix-Rosen, Martin
1001                                                              27516
1006                       932 Webster St.
1009                       2540 Pleasant St.   Raleigh            27622
1011    Mitchell, Wayne    28 Morningside Dr.  New York       NY  10017
1002                       R.R. 2, Box 1850    Hanover        NH  03755
1012    Stavros, Gloria    212 Northampton Rd. South Hadley   MA  01075
;
Run;
Proc sort data=trans;
By id;
Run;
Data newlist;
Update master trans;
By id;
Run;
```

If transaction Dataset having any missing values then output Dataset contains existing value.

```
Data ds1;
Infile Datalines;
Input id name$ sex$ sal;
Datalines;
001 abc m 5000
003 jkl f 6000
002 def m 6000
005 xyz m 5000
004 mno f 6000
006 asd f 8000
;
Run;
```

```
Proc sort Data=ds1;
By id;
Run;
Data ds2;
Infile Datalines;
Input id sal;
Datalines;
001 7500
006 9000
002 .
005 8000
004 .
003 6500
;
Run;
Proc sort Data=ds2;
By id;
Run;
Data ds3;
Update ds1 ds2;
By id;
Run;
```

Run below program and observe the difference between Merge and Update statements

Difference between Merge and Update statements

```
Data ds1;
Infile datalines;
Input a b c;
datalines;
1 2 3
4 5 6
;
Run;
Data ds2;
Infile datalines;
Input a b c;
datalines;
1 4 5
4 . 7
;
Run;
Data ds3a;
Merge ds1 ds2;
By a;
run;
Data ds3b;
update ds1 ds2;
By a;
run;
```

If transaction Dataset having any missing values then output Dataset contains missing value when we perform Merge. But when we perform Update output dataset contains existing value.

MODIFY

Replaces and appends observations in an existing SAS data set but does not create an additional dataset.

Modify statement is not required a new dataset with Data statement

The MODIFY statement avoids the creation of this temporary dataset, along with the extra temporary disk storage that it requires and the time

"Damage to the SAS data set can occur if the system terminates abnormally during a DATA step containing the MODIFY statement." So be careful.

Syntax: - Modify dataset;

Syntax: - MODIFY master-data-set <(data-set-option(s))> transaction-data-set <(data-set-option(s))>;
By Variable;

Examples:-

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

```
Data ds;
Modify ds;
If sex='F' then sex=0;
Else sex=1;
Run;
```

When we perform Modify we can't add any new Observations/ Variables and we can't delete any Observations/ Variables.

See below example and observe the difference between Modify and Set statements.

```
Data ds;
Set sashelp.class;
Run;
Data ds;
Modify ds;
If sex='F' then gender=0;
Else gender=1;
Run;
```

The same thing we can do with Set statement also but difference is

Difference between SET and MODIFY

Both are useful to modify
Set is useful to modify and can delete/add observations/variables
Modify is useful to modify existing info but can't delete/add observations/variables

```
Data ds;
Set ds;
If sex='F' then gender=0;
Else gender=1;
Run;
```

```
Data ds(drop=age);
Set ds;
If sex='F' then gender=0;
Else gender=1;
X=100;
Run;
```

Examples:-

```
Data ds1;
Infile Datalines;
Input id name$ sex$ sal;
Datalines;
001 abc m 5000
003 jkl f 6000
002 def m 6000
005 xyz m 5000
004 mno f 6000
006 asd f 8000
;
Run;
```

```
Data ds2;
Infile Datalines;
Input id sal;
Datalines;
001 7500
006 9000
002 8000
005 8000
004 7900
003 6500
;
Run;
```

```
Data ds1;
modify ds1 ds2;
Run;
```

```
ERROR: The MODIFY statement without a BY statement, or a POINT= option, or a KEY= option
requires only one data set, too many data sets have been specified.
```

Write like below and see the result.

```
Proc sort Data=ds1;
By id;
Run;
```

```
Proc sort Data=ds2;
By id;
Run;
```

```
Data ds1;
Modify ds1 ds2;
By id;
Run;
```

Difference between UPDATE and MODIFY

Both are useful to Update transaction information into master data
Update creates a new dataset and can add/delete rows/columns
But Modify updates information in master data itself it can't create a new dataset and can't add/delete rows/columns

STANSYS
SOFTWARE SOLUTIONS